TUGAS AKHIR – TE 141599

PENGEMBANGAN *SOFTWARE ECONOMIC DISPATCH* DENGAN MEMPERTIMBANGKAN *RAMP-RATE* DAN *SPINNING RESERVE* BERBASIS DELPHI

Azwar Tilameo
NRP 2211100050

Dosen Pembimbing
Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D
Feby Agung Pamuji, S.T. M.T.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2015

FINAL PROJECT – TE 141599

ECONOMIC DISPATCH SOFTWARE DEVELOPMENT CONSIDERING RAMP-RATE AND SPINNING RESERVE BASED ON DELPHI

Azwar Tilameo
NRP 2211100050

Advisor
Prof. Ir. Ontoseno Penangsang, M.Sc. Ph.D.
Feby Agung Pamuji, S.T. M.T.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Industry Technology
Sepuluh Nopember Institute of Technology
Surabaya 2015

PENGEMBANGAN *SOFTWARE ECONOMIC DISPATCH* DENGAN MEMPERTIMBANGAKAN *RAMP-RATE* DAN *SPINNING RESERVE* BERBASIS DELPHI

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Sistem Tenaga
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing I	Dosen Pembimbing II
Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D.	Feby Agung Pamuji, S.T. M.T.
NIP.194907151974121001	NIP.198702062012121002

SURABAYA
JULI, 2015

Pengembangan Software Economic Dispatch Dengan Mempertimbangkan Ramp-rate dan Spinning Reserve Berbasi Delphi

Azwar Tilameo
2211 100 050

Dosen Pembimbing 1 : Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D
Dosen Pembimbing 2 : Feby Agung Pamuji, S.T. M.T.

Abstrak:

Pembagian pembebanan pembangkit yang bertujuan meminimalkan biaya pembangkitan disebut *Economic Dispatch* (ED). Salah satu faktor/batasan dalam ED adalah *Ramp-rate* dan *Spinning Reserve*. *Software* Powergen – *software* perhitungan *economic dispatch* berbasis Delphi hasil modifikasi teknik elektro Insitiut Teknologi Sepuluh Nopember – masih belum memiliki fitur untuk mempertimbangkan batasan-batasan tersebut.

Tugas akhir ini bertujuan untuk menambahkan fitur-fitur tersebut kedalam *software* Powergen sekaligus mengetahui pengaruh penambahan batasan *ramp-rate* dan *spinning reserve* pada hasil perhitungan *economic dispatch*. Terdapat 5 kasus ED yang digunakan untuk melakukan pengujian. Di samping itu terdapat pula perhitungan manual untuk setiap kasus agar dapat dicocokkan dengan hasil perhitungan *software*. Namun perhitungan manual memakan waktu jauh lebih lama jika dibandingkan dengan menggunakan *software*.

Untuk mencapai tujuan tugas akhir ini diperlukan implementasi pada segi kode logika (*logic code*) maupun segi antar muka (*user interface*). Selanjutnya setelah selesai dikembangkan, *software* ini diuji dengan kasus pengujian yang diambil dari IEEE 18 Unit. Dari hasil pengujian terlihat bahwa hasil perhitungan *software* sudah sama dengan hasil biaya pembangkitan dari referensi [1] yaitu sebesar $101,626.3 untuk perhitungan tanpa batasan, sebesar $101,630.92 untuk perhitungan dengan batasan *ramp-rate*, dan sebesar $105,627.16 untuk perhitungan dengan batasan *ramp-rate* dan *spinning reserve*. Terlihat jelas bahwa dengan menambahkan batasan akan didapatkan biaya pembangkitan yang semakin mahal.

Kata kunci : *Software economic dispatch,* batasan *ramp-rate,* batasan *spinning reserve*, pemrograman delphi

Halaman ini sengaja dikosongkan

Economic Dispatch Software Development Considering Ramp-rate and Spinning Reserve Based on Delphi

Azwar Tilameo
2211 100 050

Advisor 1 : Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D
Advisor 2 : Feby Agung Pamuji, S.T. M.T.

Abstract:

Load-sharing across power plant that aimed at minimizing generating cost is called Economic Dispatch (ED). Some factors/constraints in ED are ramp-rate and spinning reserve. Powergen software – Delphi-based economic dispatch calculation software modified by electrical engineering Institut Teknologi Sepuluh Nopember – has not had feature to consider those constraints.

This thesis aims to add those features into Powergen software, also to determine the effect of adding ramp-rate and spinning reserve constraints on economic dispatch calculation result. There are 5 ED cases which is used to do the testing. In addition, there are also manual calculations for each case in order to be matched with the software calculation result. But the time consuming of manual calculation is much longer than using the software.

To achieve the goal of this thesis, implementation on the logic code and the user interface is required. Furthermore, once fully developed, the software is tested with test cases derived from IEEE 18 Unit. The test results show that software calculation result match the generation cost from reference [1]. The generation cost is $101,626.3 for not considering any constrain, $101,630.92 for considering ramp-rate constraint, and $105,627.16 for considering ramp-rate and spinning reserve constraint also. It shows that by adding more constraint will increase the generation cost.

Index Term : Economic dispatch software, ranp-rate constraint, spinning reserve constraint, Delphi programming.

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT atas segala rahmat, karunia, dan petunjuk yang telah dilimpahkan-Nya sehingga penulis mampu menyelesaikan tugas akhir dengan judul :

"Pengembangan *Software Economic Dispatch* Dengan Mempertimbangkan *Ramp-rate* dan *Spinning Reserve* Berbasis Delphi"

Tugas akhir ini disusun sebagai salah satu persyaratan untuk menyelesaikan jenjang pendidikan S1 pada Bidang Studi Teknik Sistem Tenaga, Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember.

Pada kesempatan ini penulis hendak menyampaikan rasa terima kasih kepada pihak-pihak yang telah memberi bantuan baik itu berupa moril maupun material, langsung maupun tidak langsung kepada:

1. Allah SWT atas limpahan Rahmat dan Petunjuk-Nya serta Nabi Muhammad SAW atas tuntunan jalan-Nya.
2. Alm. Bapak saya, Ir. Johny A. Tilameo, M.T., beserta Ibu saya , Suwarny D. Tilameo, S.K.G., yang telah membesarkan saya dan menyayangi saya serta membiayai saya hingga detik ini.
3. Prof. Ir. Ontoseno Penangsang, M.Sc. Ph.D. dan Pak Feby Agung Pamuji, S.T. M.T. sebagai dosen pembimbing yang telah memberikan arahan dan perhatiannya dalam Tugas Akhir ini.
4. Seluruh dosen yang telah memberikan ilmunya selama kuliah, karyawan, dan keluarga besar Jurusan Teknik Elektro ITS.
5. Kakak saya beserta suami dan ponakan terlucu Adelfia C. Hikari yang selalu senantiasa menemani dan memberi dukungan saat penulis berada di rumah.
6. Teman-teman Teknik Elektro ITS 2011, terutama Teman-teman satu grup Tugas Akhir atas bantuan kalian semua selama Tugas Akhir ini.
7. Teman-teman Lab Konversi Energi atas pengertiannya atas tugas akhir yang penulis tempuh.

Surabaya, Juli 2015

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Selama ini perhitungan optimalisasi sangat di butuhkan oleh perusahaan untuk mendapatkan hasil perhitungan yang paling ekonomis. Pembagian pembebanan pembangkit yang bertujuan untuk meminimalkan biaya pembangkitan disebut *Economic Dispatch* (ED). Salah satu faktor/batasan dalam ED adalah *Ramp-rate*[1] dan *Spinning Reserve*[2]. *Ramp-rate* membatasi jumlah daya yang dihasilkan oleh pembangkit yang sedang menyala. Batasan ini akan mempengaruhi pengoperasian pembangkit. [1]. Sedangkan untuk *Spinnig reserve* berdampak pada saat terjadi kegagalan pembangkit secara tiba-tiba dan diharuskan melakukan dispatch dalam waktu singkat[2]. Metode perhitungan ED saat ini sudah ada banyak sekali, mulai dari metode matematis seperti Iterasi lambda hingga metode *Artificial Intellegent* seperti *Particle Swarm Optimization*.

Meskipun sudah banyak metode penghitungan ED, aplikasi khusus yang digunakaan untuk menghitung ED masih sangat terbatas, apalagi aplikasi yang memiliki *user interface* yang baik. Oleh karena itu diperlukan sebuah aplikasi perangkat lunak yang memiliki *user interface* yang baik. Delphi merupakan sebuah bahasa pemrograman yang menawarkan *Integrated Development Environment* (IDE) visual untuk membuat program perangkat lunak berbasis sistem operasi . Delphi merupakan turunan bahasa pemrograman Pascal dan menawarkan pengembangan perangkat lunak komputer berbasis visual dengan cepat. Oleh karena itu pada tugas akhir ini akan dirancang aplikasi perhitungan ED yang mudah untuk diinteraksikan dengan pengguna menggunakan Delphi.

Teknik Elektro ITS sudah memiliki program aplikasi perhitungan ED karya sendiri berbasis Delphi yang bernama PowerGen. Program aplikasi ini memiliki beberapa fungsi seperti perhitungan ED dengan Losses, perhitungan ED dengan linear cost function, dsb. Namun belum ada fungsi yang mempertimbangkan ramp-rate dan juga spinning reserve.

1.2 Tujuan Penelitian

Penelitian pada tugas akhir ini memiliki tujuan untuk mengembangkan software perhitungan economic dispatch agar mampu mempertimbangkan pengaruh *ramp-rate* dan *spinning reserve.*

1.3 Permasalahan

Permasalahan yang akan dibahas dalam tugas akhir ini adalah :

- Apa metode optimasi *Economic Dispatch* yang dapat diterapkan pada Delphi
- Bagaimana *User Interface* yang baik dan cocok untuk perhitungan *Economic Dispatch.*
- Bagaimana kinerja(*performance*) dari software yang dihasilkan

1.4 Batasan Masalah

Agar tugas akhir ini tidak menyimpang dari ketentutan yang digariskan maka diambil batasan dan asumsi sebagai berikut :

1. Dari *software* ED yang sudah ada hanya dikembangkan perhitungan yang mempertimbangkan *Spinning Reserve* dan *Ramp-rate* tanpa memperhitungkan losses.
2. Tipe kurva pembangkit yang dibahas hanya tipe kurva polynomial orde 2 dan *piecewise incremental heat*.
3. Metode optimasi yang dibahas adalah metode iterasi lambda.

1.5 Metode Penelitian

Alur metodologi penyelesaian tugas akhir ini adalah sebagai berikut :

1. Studi pustaka
   Studi pustaka dilakukan untuk mengumpulkan buku-buku maupun jurnal yang berkaitan tentang topik tugas akhir yang dibahas. Pustaka-pustaka yang dikumpulkan mencakup *Economic Dispatch*, *Spinning Reserve* dan *Ramp-rate*, serta buku pemrograman Delphi.
2. Pengenalan *software* dan ekperimen

Pengenalan *software* dilakukan dengan mempelajari *software* yang akan dikembangkan disertai melakukan eksperimen-ekperimen untuk mengetahui bagaimana cara kerja *software* tersebut.

3. Penerapan batasan *Spinning Reserve* dan *Ramp-rate* kedalam *software*
   Dalam tahap ini *Spinning Reserve* dan *Ramp-rate* diterapkan kedalam *software* hingga dapat menghasilkan *output* dan tidak ada *error* yang terjadi.
4. Pengujian awal dan *troubleshooting* terhadap *bug* yang muncul
   Pengujian awal dilakukan untuk mencari *bug/*kesalahan dalam software. Pengujian ini lebih mengutamakan kelancaran penggunaan *software* sehingga diuji dengan data-data sederhana namun beragam.
5. Pengujian akhir dan finalisasi *software*
   Pengujian akhir dilakukan untuk memantapkan kembali kinerja *software* yang telah dikembangkan. Pada pengujian ini lebih mengutamakan kinerja software dalam melakukan perhitungan sehingga diuji dengan data-data yang lebih kompleks.
6. Pembuatan laporan tugas akhir
   Melakukan penulisan laporan yang menunjukkan hasil akhir dari tugas akhir.

1.6 Sistematika Penulisan

Sistematika penulisan laporan tugas akhir ini dibagi menjadi lima bab dengan masing-masing bab diuraikan sebagai berikut :

1. BAB 1 merupakan pendahuluan yang berisi latar belakang, permasalahan, tujuan, metodologi, batasan masalah dan sistematika penulisan.
2. BAB 2 berisi teori penunjang yang membahas tentang Sistem kelistrikan, *Economic Dispatch*, batasan *Spinning Reserve* dan *Ramp-rate*, serta dasar-dasar permrograman Delphi
3. BAB 3 berisi tentang uraian perencanaan, pembuatan, dan implementasi kedalam *software* yang dikembangkan.
4. BAB 4 berisi tentang hasil pengujian perangkaat lunak yang telah dirancang.
5. BAB 5 berisi tentang kesimpulan dan saran-saran dari pembuatan sampai pengimplementasian perangkat lunak.

1.7 Manfaat dan Relevansi

1. Bagi perusahaan listrik
   Tugas akhir ini diharapkan dapat memberikan manfaat bagi perusahaan listrik dalam memutuskan pola pembangkitan yang dilakukan sehingga mendapatkan biaya pembangkitan yang lebih baik.
2. Bagi bidang ilmu pengetahuan dan mahasiswa lain
   Tugas akhir ini diharapkan dapat membantu perkembangan ilmu pengetahuan dengan menjadi alat bantu perhitungan ED yang handal dan mudah digunakan.

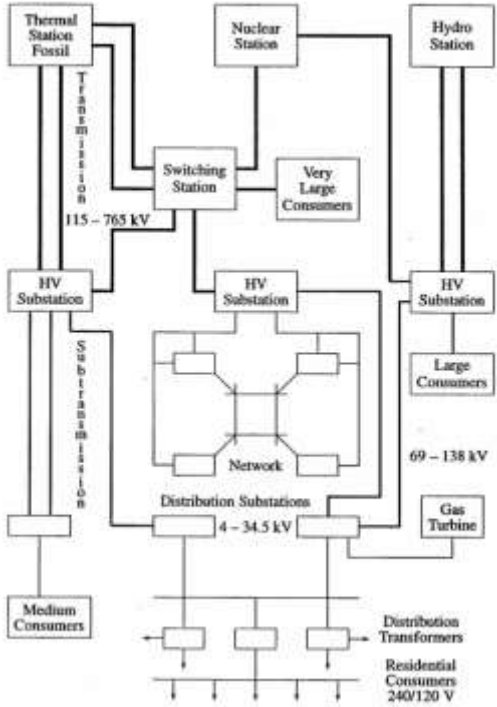
BAB 2
TEORI PENUNJANG

2.1 Sistem Tenaga Listrik

Energi listrik adalah bentuk energi yang paling populer karena bentuk energi ini dapat dipindahkan dengan mudah pada efisiensi yang tinggi serta biaya yang masuk akal.

Sistem tenaga listrik adalah sebuah sistem yang bertujuan untuk memanfaatkan energi listrik. Komponen-komponen sistem tenaga listrik terbagi menjadi tiga sistem, yaitu: sistem pembangkitan; sistem transmisi; dan sistem distribusi. Sistem tenaga listrik secara umum digunakan untuk memenuhi kebutuhan energi listrik. Sistem pembangkitan terdiri dari kumpulan unit pembangkit tenaga listrik yang terhubung dengan sistem transmisi dan sistem distribusi. Sistem transmisi digunakan dalam penyaluran daya listrik dari sistem pembangkit menuju sistem distribusi dengan menggunakan tegangan tinggi untuk mengurangi rugi-rugi saluran. Sedangkan untuk sistem distribusi terdiri dari gardu induk dan beban[5].

Sistem tenaga listrik modern terdiri dari beberapa pembangkit yang saling terhubung. Sistem tenaga listrik yang seperti ini disebut juga dengan sistem tenaga listrik terinterkoneksi[6]. Keuntungan dari penggunaan sistem tenaga listrik terinterkoneksi antara lain: dapat meningkatkan keandalan sistem; meningkatkan efisiensi pembangkit; dapat menyalurkan daya listrik ke daerah yang jauh dari sistem pembangkitan, dan mempermudah penjadwalan pembangkit. Contoh sistem tenaga listrik modern dapat dilihat pada gambar 2.1



Gambar 2.1 Sistem tenaga listrik modern[4]

2.1.1 Sistem Pembangkitan

Secara umum sistem pembangkitan merupakan kumpulan dari unit pembangkit tenaga listrik yang terdiri dari beberapa komponen utama seperti turbin dan generator. Pembangkit tenaga listrik digunakan untuk membangkitkan daya listrik yang kemudian didistribusikan kepada konsumen. Di dalam sebuah sistem pembangkit, beberapa generator dioperasikan secara paralel dan dihubungkan dengan bus dalam suatu sistem tenaga listrik guna menyediakan total daya yang diperlukan[4]

Pembangkit tenaga listrik dapat dibedakan menjadi beberapa jenis sesuai dengan bahan bakar yang digunakan. Salah satu diantaranya adalah pembangkit listrik tenaga panas atau *thermal*. Pembangkit tipe ini

merupakan pembangkit listrik yang mayoritas digunakan untuk memenuhi beban harian atau *base load*.

Setiap pembangkit memiliki karaktersitik unit pembangkit masing-masing. Karakteristik unit pembangkit meliputi karaktersitik *input-output* pembangkit, dan karakteristik *incremental rate*[7]. Karakteristik tersebut diperoleh dari data-data seperti : desain generator; pabrik pembuat generator; data historis pengoperasian generator; maupun data percobaan. Karakteristik unit pembangkit digunakan dalam perhitungan biaya pembangkitan dari tiap unit pembangkit sehingga dapat dicapai nilai ekonomis atau nilai optimum.

Karakteristik *input-output* dari pembangkit dari pembangkit *thermal* merupakan hubungan antara *input* berupa bahan bakar yang digunakan dengan *output* berupa daya yang dibangkitkan tiap pembangkit. *Input* bahan bakar dinyatakan dalam bentuk MBtu/h atau konsumsi energi sedangkan output daya dinyatakan dalam bentuk MW atau daya yang dibangkitkan.

Karakteristik *incremental rate* pembangkit thermal merupakan hubungan antara perubahan daya pembangkitan yang dihasilkan dengan konsumsi bahan bakar yang dibutuhkan. *Incremental rate* biasanya dinyatakan dengan satuan Btu/kWh. Penjelasan mengenai karakteristik pembangkit akan dibahas lebih lanjut pada sub-bab 2.2

2.1.2 Sistem Transmisi

Transmisi pada sistem tenaga listrik merupakan jaringan listrik yang berfungsi untuk menyalurkan daya listrik yang dibangkitkan melalui unit pembangkit tenaga listrik menuju sistem sistem distribusi. Pada sister transmisi tegangan yang digunakan adalah Tegan[gan tinggi guna mengurangi rugi jaringan atau *losses* transmisi yang disebabkan oleh panas penghantar akibat adanya arus yang mengalir[4].

Jaringan transmisi yang saling terinterkoneksi, selain dapat digunakan untuk melakukan operasi yang ekonomis antar pembangkit dapat pula digunakan untuk saling mengirimkan energi pada saat darurat.

Gambar 2.1 menunjukkan diagram transmisi dan distribusi sistem tenaga listrik. Saluran transmisi tegangan tinggi bermuara pada gardu induk (Substation). Gardu induk merupakan salah satu komponen sistem transmisi yang digunakan sebagai suatu tempat persinggahan energi listrik.

Saluran pada sistem transmisi dapat dibedakan menurut besar tegangannnya yaitu sebagai berikut:

1. Saluran Udara Tegangan Ekstra Tinggi (SUTET): 200kV-500kV
2. Saluran Udara Tegangan Tinggi (SUTT): 30kV – 150kV
3. Saluran Kabel Tegangan Tinggi (SKTT): 30kV – 150kV

2.1.3 Sistem Distribusi

Sistem distribusi berfungsi untuk menyalurkan daya listrik dari sistem transmisi melalui gardu induk distribusi ke peralatan konsumen. Sistem distribusi dapat dibagi berdasarkan jenis pelanggannya yaitu sistem distribusi primer dan distribusi sekunder.

Sistem distribusi primer digunakan untuk malayani pelanggan dengan kapasitas daya besar seperti sektor industri. Tegangan yang digunakan pada sistem distribusi primer berkisar antara 2 kV sampai dengan 34.5 kV. Sedangkan untuk sistem distribusi sekunder digunakan untuk melayani pelanggan residensial dan komersial. Tegangan yang digunakan pada sistem distribusi sekunder adalah tegangan satu phasa 220/120 V dengan tiga kawat, tegangan tiga phasa 220/120 V dengan empat kawat, tegangan tiga phasa 480/277 V dengan empat kawat[4].

2.2 Karakteristik Unit Pembangkit

Perbedaan karakteristik unit pembangkit menyebabkan setiap unit pembangkit memiliki porsi yang berbeda-beda dalam mensuplai beban suatu sistem tenaga listrik. Secara umum pembangkit digolongkan menjadi tiga yaitu pembangkit beban dasar (*base load*), pembangkit beban menengah (*load follower*) dan pembangkit beban puncak (*peaker*).

Pembangkit dengan karakteristik yang kurang fleksibel – pembangkit yang tidak dapat dihidupkan dan dimatikan dalam waktu singkat – megharuskan dirinya untuk dioperasikan sepanjang pembangit siap. Pembangkit yang seperti ini digolongkan ke dalam pembangkit *base load*. Pembangkit *base load* berskala besar dan memiliki biaya produksi yang lebih murah dibandingkan dengan pembangkit jenis lain. Contoh pembangkit jenis ini adalah PLTU (Pembangkit Listrik Tenaga Uap) batubara atau pembangkit hidro dengan sumber air yang hanya akan ekonomis bila dioperasikan (tipe *run off river*)[4].

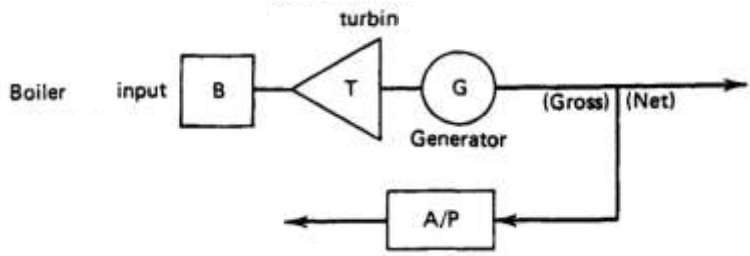
Pembangkit kelompok *load follower* meliputi pembangkit yang lebih fleksibel namun lebih mahal dari pembangkit *base load*. Contoh pembangkit jenis ini adalah PLTGU (Pembangkit Listrik Tenaga Gas-Uap) gas dan PLTU minyak.

Pembangkit golongan yang terakhir adalah pembangkit yang difungsikan sebagai pemikul beban puncak (*peaker*). Pembangkit golongan ini meliputi pembangkit yang fleksibel. Fleksibel baik dalam kecepatan perubahanpembebanan maupun operasi hidup dan mati pembangkit. Pembangkit jenis ini rata-rata berkapasitas di bawah 100 MW. Contoh dari pembangkit jenis ini adalah PLTG (Pembangkit Listrik Tenaga Gas) minyak, PLTD serta PLTA waduk.

2.2.1 Pemodelan Unit Pembangkit

Dasar perhitungan operasi ekonomis adalah pada karakteristik input-output unit-unit pembangkitnya. Pada pembangkit *thermal*, karakteristik *input-output* konsumsi bahan bakar pembangkit merupakan dasar penyusunan fungsi biaya.

Pembangkit thermal sederhana – seperti pada Gambar 2.2 – terdiri dari boiler, turbin uap dan generator. *Input* boiler adalah bahan bakar dan output berupa uap. *Input* dari turbin-generator berupa uap dan *output* nya berupa daya listrik. Karakteristik dari keseluruhan sistem suatu pembangkit dapat diekspresikan secara langsung dengan menggabungkan karakteristik input-output dari boiler dan turbin-generator.



Gambar 2.2 Pemodelan boiler-turbin-generator[7]

Persamaan karakteristik input-output pembangkit *thermal* secara umum direpresentasikan dalam persamaan orde dua. Tetapi persamaan tersebut dapat juga memiliki orde lebih dari dua, atau bahkan bisa juga

menjadi lebih tidak linear (*non-convex*) apabila memperhatikan pengaruh-pengaruh kecil seperti *valve-point effect*.

Untuk menganalisis permasalahan mengenai operasi dalam sistem tenaga, terutama masalah operasi ekonomis, dibutuhkan dasar mengenai karakteristik *input-output* dari suatu unit pembangkit *thermal*. Untuk mendefinisikan karakteristik unit, dibutuhkan penjelasan mengenai *gross input* dan *net output*[7].

Gross input dari suatu pembangkit merepresentasikan total input dan diukur dalam dolar per jam($/jam) atau kubik gas per jam(gas^3/jam) atau bentuk lain. Sedangkan *net output* dari suatu pembangkit adalah output daya listrik yang tersedia untuk penggunaan pada sistem tenaga.

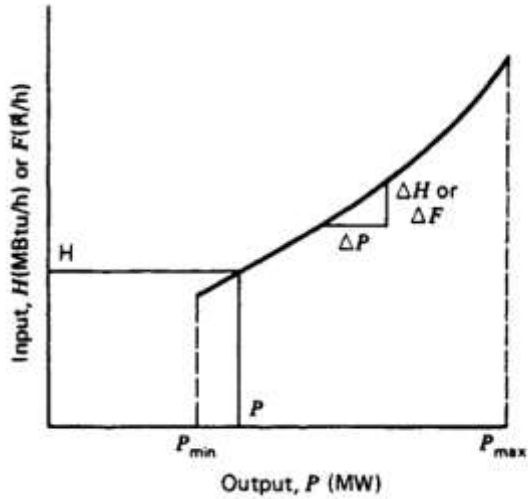
Dalam mendefinisikan karakteristik dari unit turbin uap digunakan beberapa istilah sebagai berikut :

H = Besaran panas sebagai *input* unit pembangkit (MBtu/jam)
F = Besaran biaya input unit pembangkit ($harga\ bahan\ bakar \times H$) ($/jam)

Biaya operasional $ per jam suatu unit terdiri atas biaya operasional dan biaya pemeliharaan. Sehingga biaya pekerja akan dimasukkan sebagaibagian dari biaya operasi jika biaya ini dapat digambarkan secara langsung sebagai fungsi dari output unit. Output dari unit pembangkit dinotasikan dengan P.

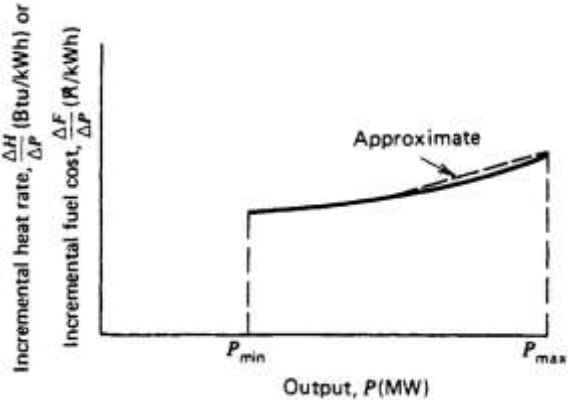
Seperti yang sudah dijelaskan sebelumnya, setiap pembangkit memiliki karakteristik *input-output* dan karakteristik *incremental rate (*baik *incremental heat* maupun *incremental cost*).

Gambar 2.3 menunjukkan karakteristik *input-output* dari suatu unit pembangkit thermal. Karakteristik *input-output* dari pembangkit dari pembangkit *thermal* merupakan hubungan antara *input* berupa bahan bakar yang digunakan dengan *output* berupa daya yang dibangkitkan tiap pembangkit. *Input* bahan bakar dinyatakan dalam bentuk MBtu/h atau konsumsi energi sedangkan output daya dinyatakan dalam bentuk MW atau daya yang dibangkitkan.



Gambar 2.3 Karakteristik *input-output* unit pembangkit[7]

Karakteristik *incremental rate* pembangkit thermal merupakan hubungan antara perubahan daya pembangkitan yang dihasilkan dengan konsumsi bahan bakar yang dibutuhkan. *Incremental rate* menunjukkan seberapa besar biaya/panas yang harus ditambahkan saat akan meningkatkan *output* unit pembangkit tersebut. *Incremental rate* sebenarnya menyatakan gradient/kemiringan kurva *input-output*. *Incremental rate* biasa dinyatakan dengan simbol $\Delta H/\Delta P$ – atau lebih dikenal dengan sebutan IHR (*incremental heat rate)* – memiliki satuan Btu/kWh. Contoh kurva karakteristik *incremental rate* dapat dilihat pada gambar 2.4



Gambar 2.4 Karakteristik *incremental rate*[7]

Selanjutnya akan dibahas lebih lanjut tentang pemodelan karakteristik *input-output* maupun karakteristik *incremental rate*. Ada dua macam pendekatan dalam memodelkan karakteristik-karakteristik tersebut. Yang paling umum ditemui adalah pemodelan dengan fungsi polinomial. Namun disamping itu tidak jarang juga kita temui bentuk fungsi *piecewise*

2.2.1.1 Pemodelan Fungsi Polinomial (Continuous)

Bentuk pemodelan fungsi polinomial adalah pendekatan dari kurva *input-output* dengan fungsi polinomial. Fungsi polinomial yang umum digunakan adalah kurva polinomial orde dua. Namun meski begitu tidak menutup kemungkinan bila nantinya ada pendekatan dengan fungsi polinomial dengan orde lebih dari dua.

Sebagai contoh dari gambar 2.3 kita dapat membuat pendekatan fungsi polinomial orde dua seperti pada persamaan 2.1

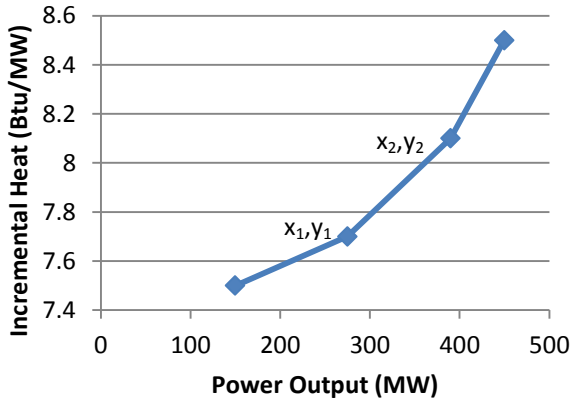
$$\boldsymbol{H(P) = aP^2 + bP + c} \tag{2.1}$$

Sedangkan fungsi *incremental rate* nya bisa kita dapatkan dari turunan pertama fungsi *input-output*.

$$\boldsymbol{ihr(P) = \frac{\delta H(P)}{\delta P} = 2aP + b} \tag{2.2}$$

2.2.1.2 Pemodelan Piecewise Incremental Heat

Dalam ilmu matematika, fungsi *piecewise* adalah fungsi yang didefinisikan oleh sub fungsi yang digunakan pada interval/segmen yang berbeda. Pemodelan bentuk ini menyajikan serangkaian set data dari kurva *incremental* heat yang kemudian dapat kita definisikan ke dalam bentuk polinomial untuk setiap interval/segmen. Penjelasannya akan lebih mudah jika kita mengamati Gambar 2.5



Gambar 2.5 Contoh kurva *piecewise incremental rate*

Untuk segmen antara titik (x_1,y_1) dan (x_2,y_2) dapat kita bentuk persamaan polinomialnya

$$\boldsymbol{ihr}(\boldsymbol{P}) = \propto \boldsymbol{P} + \boldsymbol{\beta} \tag{2.3}$$

Dimana

$$\propto = \frac{\boldsymbol{y_2 - y_1}}{\boldsymbol{x_2 - x_1}} \tag{2.4}$$

$$\boldsymbol{\beta} = \frac{\boldsymbol{x_2 y_1 - x_1 y_2}}{\boldsymbol{x_2 - x_1}} \tag{2.5}$$

Sehingga persamaannya menjadi

$$\boldsymbol{ihr(P) = \frac{y_2 - y_1}{x_2 - x_1} P + \frac{x_2 y_1 - x_1 y_2}{x_2 - x_1}} \quad (2.6)$$

Dari persamaan 2.6 kita bisa mendapatkan fungsi *input-output* nya dengan mengintegralkan fungsi ihr.

$$\boldsymbol{H = \int ihr(P)\, dp = \frac{1}{2} \propto P^2 + \beta P + C} \quad (2.7)$$

Dimana C adalah bahan bakar minimum saat output masih nol megawatt. C disebut juga *no load fuel*, atau pada fungsi biaya C disebut *no load cost*.

2.3 Economic Dispatch

Economic Dispatch (ED) adalah pembebanan pada pembangkit-pembangkit yang ada dalam sistem secara optimal dari sisi ekonomi dengan beban tertentu. Besar beban pada suatu sistem tenaga selalu berubah setiap periode waktu tertentu, oleh karena itu untuk mensuplai beban secara ekonomis maka perhitungan ED dilakukan pada setiap beban tersebut.

Pada pembangkitan energi listrik, terdapat tiga komponen biaya utama. Biaya biaya tersebut antara lain biaya pembangunan fasilitas, biaya kepemilikan, dan biaya operasi. Biaya operasi adalah biaya yang memiliki bagian yang paling dominan pada sistem operasi tenaga listrik[9].

Salah satu komponen dominan pada biaya operasi adalah biaya bahan bakar (*fuel cost*) dan setiap pembangkit memiliki karakteristik *fuel cost* yang berbeda-beda sesuai dengan jenis bahan bakar dan efisiensi dari pembangkit. Pengoptimalan biaya operasi dengan mempertimbangkan *fuel cost* sangat mempengaruhi biaya produksi energi listrik. Oleh karena itu – meskipun dalam kondisi operasi normal – penjadwalan kerja tiap pembangkit tetap menjadi prioritas utama untuk menekan biaya produksi.

Tujuan utama dari *Economic Dispatch* adalah untuk meminimalkan konsumsi bahan bakar dari pembangkit pada keseluruhan sistem dengan menentukan daya output setiap pembangkit. Penentuan daya output pada setiap generator hanya boleh bervariasi pada batasan (*constrain*) tertentu.

Permasalahan ED merupakan permasalahan pengoptimalan yang rumit. Proses pengoptimalannya dengan mengoptimasikan biaya bahan

baku pembangkitan (*fuel cost*) – yang memiliki karakteristik tidak linear – agar minimum. Seperti telah dijelaskan sebelumnya, bentuk tipikal dari persamaan biaya pembangkit direpresentasikan dengan fungsi orde dua seperti pada Persamaan (2.1). Dalam pembahasan ED, tiap unit ke 'i' akan memiliki persamaan fungsi biaya seperti berikut

$$\boldsymbol{F_i(P_i) = a_i P_i^2 + b_i P_i + c_i} \tag{2.7}$$

Dimana :

$F_i =$ Besar biaya pembangkitan pada unit ke-i ($)
$P_i =$ Daya *output* dari pembangkit ke-i (MW)

Variable $a, b,$ dan c adalah koefisien biaya operasi produksi dari suatu pembangkit. Koefisien c juga merepresentasikan biaya operasi pembangkit ketika tidak memproduksi energi listrik.

Dari Persamaan (2.7), dapat diketahui bahwa hubungan antara daya yang dibangkitkan dari generator tidak linear terhadap biaya pembangkitan. Kombinasi daya output yang dibangkitkan oleh tiap-tiap generator pada sistem harus memenuhi kebutuhan daya dari sistem tenaga listrik dan memenuhi batas minimum serta maksimum dari daya yang dapat dibangkitkan oleh generator. Karena permasalahannya rumit, maka permasalahan ED hanya bisa dilakukan dengan metode iterasi. Parameter-parameter yang telah dijelaskan diatas dapat direpresentasikan dalam Persamaan (2.8 – 2.10)

$$\boldsymbol{\min(F_i) = \min\left(\sum(a_i P_i^2 + b_i P_i + c_i)\right)} \tag{2.8}$$

$$\boldsymbol{P_i^{min} \leq P_i \leq P_i^{max}} \tag{2.9}$$

$$\boldsymbol{P_L = \sum P_i - P_{loss}} \tag{2.10}$$

Dimana :

P_i = Daya terbangkit pada unit ke-i
P_L = Total daya beban
P_{loss} = Total daya yang hilang akibat rugi transmisi

Persamaan (2.10) adalah persamaan yang memperhitungkan rugi transmisi. Besarnya daya yang dibangkitkan oleh unit pembangkit akan

mempengaruhi besarnya rugi transmisi. Namun pada tugas akhir ini rugi transmisi akan diabaikan.

Karena pada tugas akhir ini rugi transmisi diabaikan, total daya beban merupakan jumlah dari total seluruh pembangkitan. Permasalahannya adalah untuk mencari daya yang dibangkitan oleh setiap unit pembangkit dengan *objective function* (total biaya produksi) seperti pada persamaan (2.8) yang minimum dengan terbatas pada persamaan (2.9). Sedangkan persamaan (2.10) berubah menjadi

$$\boldsymbol{P_L = \sum P_i} \tag{2.11}$$

Salah satu pendekatan konvensional menggunakan persamaan *Lagrange* (*Lagrange multiplier*)

$$\boldsymbol{\mathcal{L} = F_t + \lambda(P_L - \sum P_i)} \tag{2.12}$$

Dimana F_t adalah total biaya produksi dari seluruh unit pembangkit yang ada. Nilai minimum dari persamaan (2.12) akan didapatkan saat turunan parsial terhadap daya yang dibangkitkan sama dengan nol[4].

$$\boldsymbol{\frac{\partial \mathcal{L}}{\delta P_i} = 0} \tag{2.13}$$

$$\boldsymbol{\frac{\partial \mathcal{L}}{\partial \lambda} = 0} \tag{2.14}$$

Dengan menggabungkan persamaan (2.13) dengan persamaan (2.12), serta karena

$$\boldsymbol{F_t = F_1 + F_2 + \cdots + F_n} \tag{2.15}$$

maka

$$\boldsymbol{\lambda = \frac{dF_i}{dP_i}} \tag{2.16}$$

$$\boldsymbol{\lambda = 2aP_i + b} \tag{2.17}$$

Untuk beroperasi secara ekonomis, maka seluruh unit pembangkit harus beroperasi pada nilai *incremental cost* yang sama. Selain itu

tentunya memenuhi batasan-batasan *economic dispatch* yang lainnya, terutama batasan pada persamaan (2.11). Pembangkitan untuk setiap unit dapat dihitung dengan persamaan

$$\boldsymbol{P_i = \frac{\lambda - b_i}{2a_i}} \tag{2.18}$$

Persamaan (2.18) dikenal dengan nama persamaan koordinasi. Persamaan tersebut merupakan fungsi dari lambda. Solusi analitis untuk mendapatkan nilai lambda bisa kita peroleh dengan mensubtitusi persamaan (2.18) dengan persamaan (2.11)

$$\boldsymbol{\sum_{i=1}^{n} \frac{\lambda - b_i}{2a_i} = P_L} \tag{2.19}$$

$$\boldsymbol{\lambda = \frac{P_L + \sum \frac{b_i}{2a_i}}{\sum \frac{1}{2a_i}}} \tag{2.20}$$

2.3.1 Batasan Ramp-rate

Batasan *ramp rate* muncul ketika engineer pada masa itu memasukkan faktor *valve throttling* pada pembangkit thermal kedalam batasan perhitungan economic dispatch [3]. *Ramp-rate Limit* adalah batasan perubahan daya terbangkitkan sebuah pembangkit tenaga listrik. *Ramp-rate Limit* dibagi menjadi 2 jenis yaitu *Up-Rate* (UR) dan *Down-Rate* (DR) [1].

Pada saat geneator menambah pembangkitan, maka batasannya

$$\boldsymbol{P_{it} - P_{i(t-1)} \leq UR_i} \tag{2.21}$$

Sedangkan saat generator mengurangi pembangkitan, maka batasannya

$$\boldsymbol{P_{i(t-1)} - P_{it} \leq DR_i} \tag{2.22}$$

Sehingga batasan generator untuk perhitugan mempertimbangakn *ramp-rate limit* menjadi

$$\max\left(P_{imin}, P_{i(t-1)} - DR_i\right) \leq P_{it} \leq \min(P_{imax}, P_{i(t-1)} + UR_i) \quad (2.23)$$

2.3.2 Batasan Spinning reserve

Spinning Reserve / cadangan berputar merupakan salah satu faktor lainnya yang biasa menjadi batasan dalam ED. Faktor ini didefinisikan sebagai total pembangkitan yang tersedia dari seluruh pembangkit yang aktif. *Spinning Reserve* (SR) harus diperhatikan agar ketika terjadi kegagalan pada salah satu pembangkit tidak terlalu mempengaruhi frekuensi sistem [2]. Batasan SR dalam perhitungan ED akan berpengaruh pada batas pembangkitan yang diperbolehkan seperti berikut

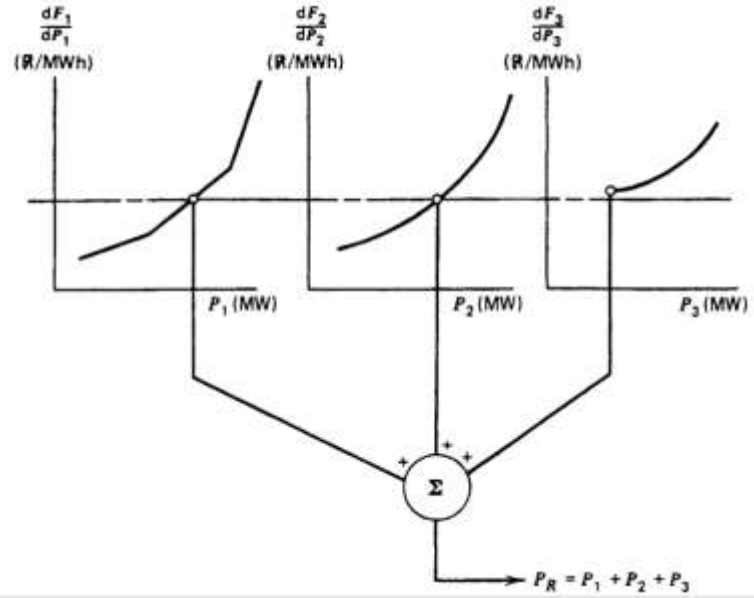
$$P_i^{\min} \leq P_i + SR_i \leq P_i^{\max} \quad (2.24)$$

2.4 Metode Iterasi Lambda

Konsep dari iterasi lambda adalah dengan mengamati karakteristik *incremental heat rate* (ihr) dari setiap unit pembangkit. Gambar 2.6 menunjukkan bagaimana konsep iterasi lambda bekerja. Lambda yang dimaksud pada bahasan ini adalah nilai dari *incremental heat rate*.

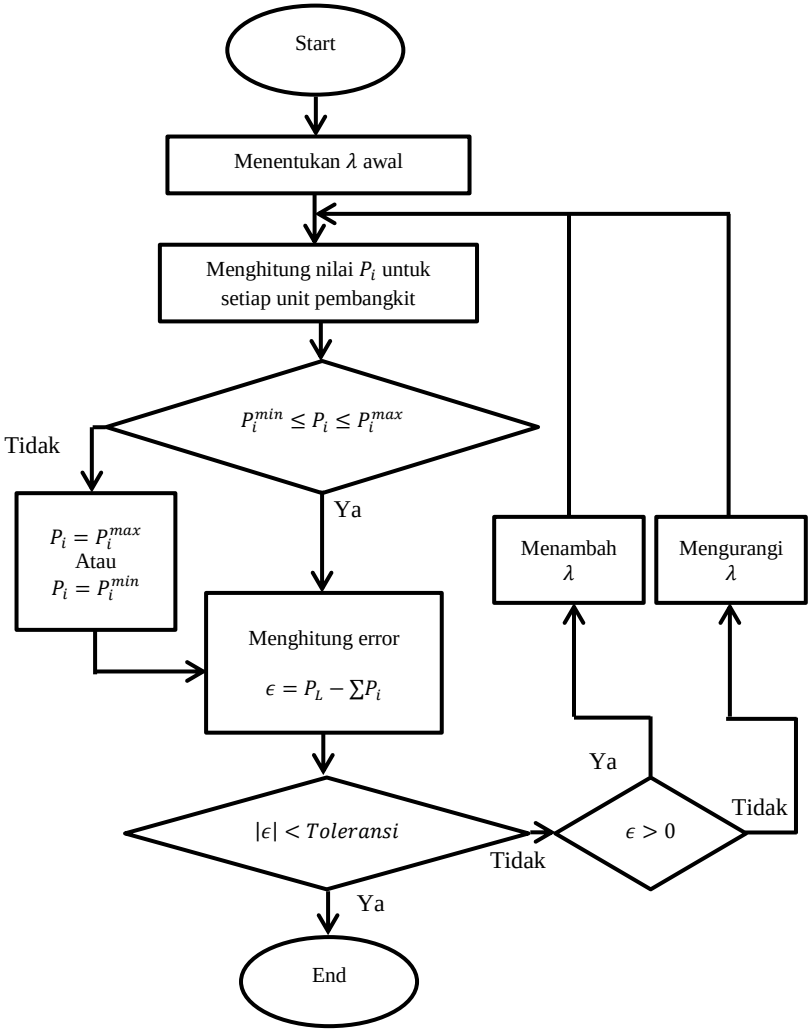
$$\lambda = ihr = 2a\,P + b \quad (2.25)$$

Kemudian berdasarkan kurva karakteristiknya dapat diketahui nilai daya terbangkit yang dihasilkan untuk setiap nilai lambda. Nilai lambda ini di iterasi hingga menghasilkan nilai daya terbangkit yang memenuhi total daya beban.



Gambar 2.6 Konsep iterasi lambda[7]

Penjelasan lebih rinci mengenai metode ini bisa dilihat pada *flowchart* Gambat 2.7.



Gambar 2.7 *Flowchart* metode iterasi lambda

Metode ini diawali dengan menentukan nilai lambda awal. Dari nilai lambda tadi kemudian dihitung nilai pembangkitan yang sesuai

dengan nilai lambda tersebut. Apabila nilai pembangkitan melanggar batas pembangkitan pembangkit – yang dapat berupa Persamaan (2.9), Persamaan (2.23) ataupun Persamaan (2.24) – maka nilai P_i sama dengan nilai maksimum/minimum nya. Selanjutnya total nilai daya pembangkitan dihitung dan dibandingkan dengan nilai daya beban untuk mengetahui errornya. Apabila error masih lebih besar dari toleransi, maka nilai lambda (λ) diubah sesuai besarnya error tersebut. Iterasi selanjutnya dilanjukan kembali untuk nilai lambda (λ) yang baru.

2.5 Delphi

Pada awalnya Delphi adalah proyek rahasia di Borland yang berevolusi menjadi sebuah produk yang disebut AppBuilder. Seiring perkembangan, proyek ini mendapatkan popularitas di kalangan tim *developer* dan kelompok *beta-testing* dan akan diberi nama "Borland AppBuilder". Sesaat sebelum rilis pertama dari Borland, Novell AppBuilder dirilis sehingga Borland harus memberikan nama baru untuk proyek tersebut. Namun pada akhirnya produk tersebut dirilis dengan nama Delphi[11].

Dewasa ini perkembangan *Software Development Tool* sangat pesat dengan persaingan yang sangat ketat. Masing-masing perangkat memiliki kelebihan dan kekurangan yang sangat tipis bila dibandingkan satu dengan yang lainnya.

Kelebihan Delphi yang pertama adalah kemudahan penyusunan *user interface*. Sejak awal dirilis, Delphi berkomitmen untuk menjadi *Rapid Application Development* (RAD) *Tool*. Maksudnya adalah bagaimana menjadi pernagkat yang mempercepat pengembangan aplikasi. Untuk RAD ini, Delphi telah melakukannya dengan sangat baik, dimulai dari kemudahan penyusunan tampilan program.

Selain itu, dengan konsep komponen yang sangat terbuka, Delphi telah memungkinkan banyak pihak untuk mengembangkan berbagai macam komponen untuk berbagai macam kebutuhan.

Bahasa yang digunakan dalam Delphi adalah Object Pascal dengan sejumlah penambahan, terutama terkait dengan konsep *Object Oriented Programming* (OOP).

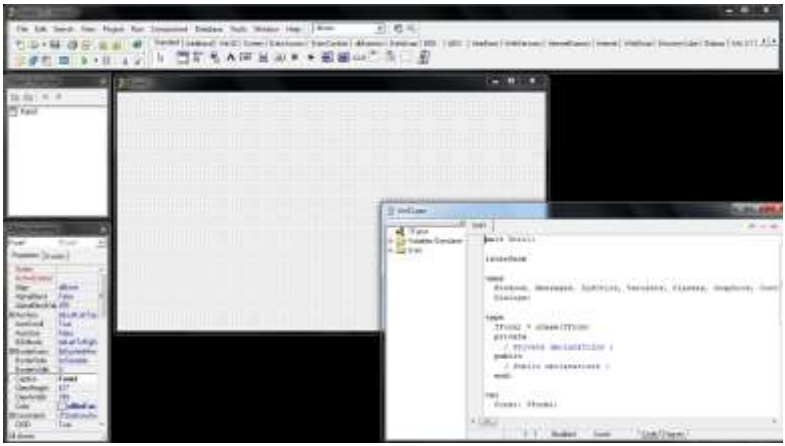
Salah satu kelebihan bahasa Pascal adalah mudah untuk dipelajari. Sejak awal dibuat, bahasa ini sudah digunakan untuk pengajaran bahasa pemrograman di banyak perguruan tinggi di seluruh dunia. Object Pascal

pada Delphi adalah salah satu versi Pascal yang sangat *powerful*, tetapi tidak menjadikannya terlalu kompleks.

Dengan kelebihannya ini, Delphi bisa menjadi alat yang cukup baik untuk memahami konsep-konsep penting dalam pemrograman, seperti algoritma dan struktur datam serta konsep *Object Oriented Programming*, Pemahaman yang baik terhadap algoritma dan struktur data serta konsep OOP yang telah kita oelajari akan tetap melekat, dan tetap dapat digunakan ketika kita menggunakan bahasa pemrograman yang lain.

2.5.1 Delphi IDE (*Integrated Development Environment*)

Saat pertama kali menjalan Delphi, kita akan dihadapkan dengan tampilan IDE (*Integrated Development Environment*) seperti pada Gambar 2.8. IDE menyediakan seluruh *tools* yang kita butuhkan untuk mendesain, mengembangkan, menyelesaikan *bug*, maupun menguji aplikasi yang kita buat[12].



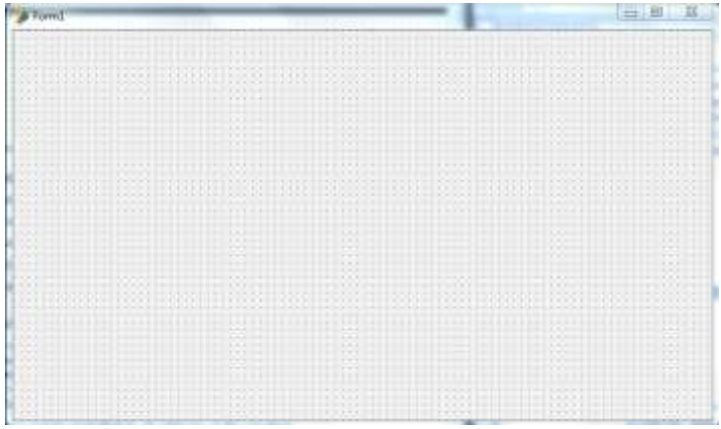
Gambar 2.8 Tampilan *Integrated Development Environment* Delphi

IDE menyediakan *tools* yang akan kita butuhkan untuk memulai mendesai sebuah aplikasi. *Tools* tersebut antara lain :

- *Form Designer* (Gambar 2.9), jendela kosong tempat dimana kita mendesain *user interface* (UI) untuk aplikasi kita.

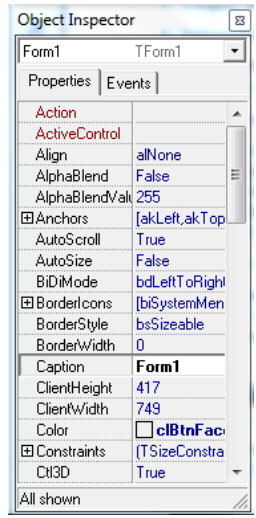
- *Component Pallete* (Gambar 2.10) untuk memunculkan komponen-komponen visual maupun non-visual yang kita gunakan untuk mendesain (UI).
- *Object Inspector* (Gambar 2.11) untuk memeriksa dan mengatur *properties* serta *event* dari sebuah objek.
- *Object Tree View* (Gambar 2.12) untuk memunculkan dan mengubah-ubah hubungan antar komponen.
- *Code Editor* (Gambar 2.13) untuk menulis kode-kode logika program.
- *Project Manager* (Gambar 2.14) untuk mengatur *file-file* dan menjadikannya dalam satu proyek.
- *Integrated Debugger* untuk menjadi dan membenarkan error dalam kode-kode kita.
- *Command-line tools* seperti *compiler*, *linker*, dan yang lainnya.
- *Library* yang luas dengan objek-objek yang dapat digunakan kembali

.

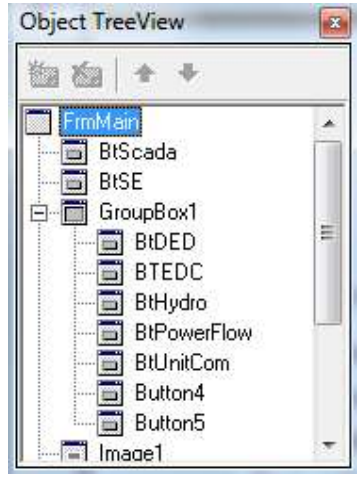


Gambar 2.9 Tampilan *Form Designer*

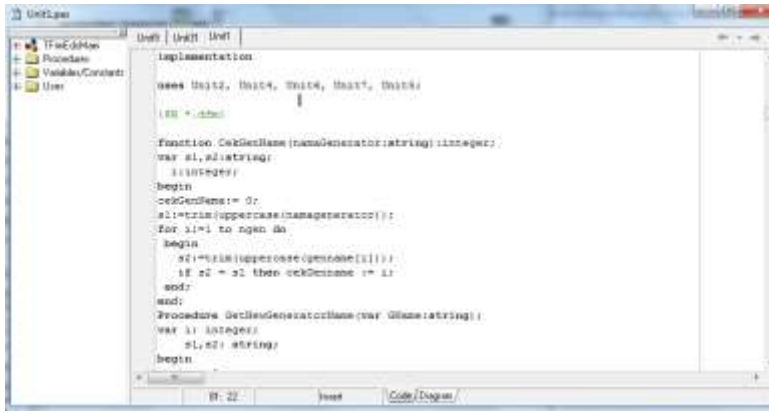
Gambar 2.10 Tampilan *Component Pallete*



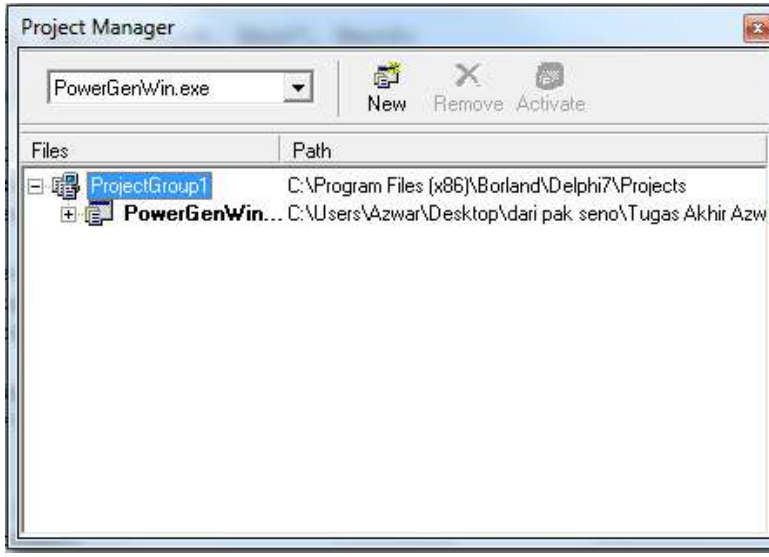
Gambar 2.11 Tampilan *Object Inspector*



Gambar 2.12 Tampilan *Object Treeview*



Gambar 2.13 Tampilan *Code Editor*



Gambar 2.14 Tampilan *Project Manager*

Halaman sengaja dikosongkan

BAB 3
RANCANGAN PENGEMBANGAN *SOFTWARE*

3.1 *Software* Powergen

Software Powergen adalah sebuah perangkat lunak milik Teknik Elektro Institut Teknologi Sepuluh Nopember yang digunakan untuk melakukan perhitungan-perhitungan yang berkaitan dengan Teknik Sistem Tenaga. *Software* ini sudah digunakan sebagai alat bantu pada proses akademik. Salah satu mata kuliah yang menggunakan *software* ini adalah mata kuliah operasi optimum. Tampilan menu utama *software* Powergen dapat dilihat pada Gambar 3.1.

Gambar 3.1 Tampilan Menu Utama *Software* Powergen

Software Powergen yang akan dikembangkan memiliki beberapa fitur antara lain :

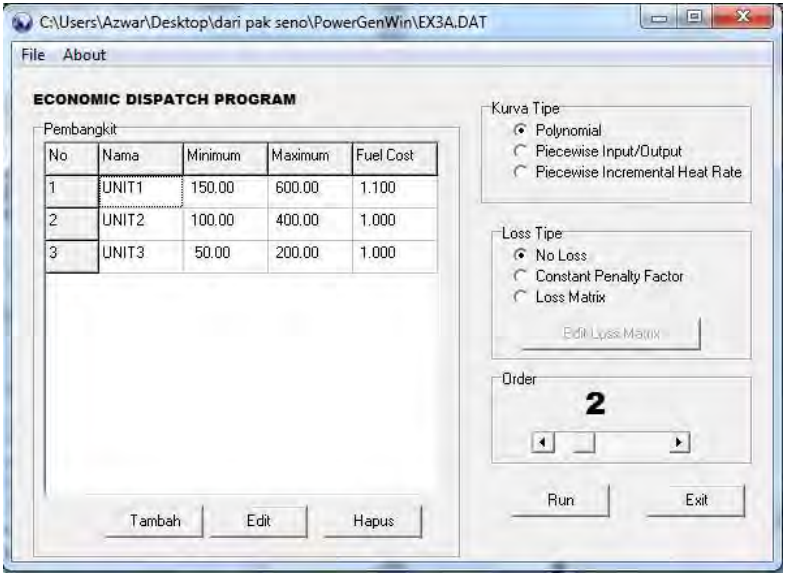
- Power Flow : Melakukan perhitungan aliran daya
- DUBLP : Melakukan perhitungan optimasi sederhana dengan *linear programming*
- EDC : Melakukan perhitungan *economic dispatch*

- Hydro : Menyelesaikan permasalahan penjadwalan pembangkit tenaga air/*hydro*
- Unitcom : Melakukan perhitungan *Unit Commitment*
- DED : Melakukan perhitungan Dynamic Economic Dispatch

Pada tugas akhir ini menu yang akan dikembangkan adalah menu EDC (*Economic Dispatch*) dan DED (*Dynamic Economic Dispatch*). Oleh karena itu akan dijelaskan terlebih dahulu fitur-fitur yang telah ada pada kedua menu tersebut.

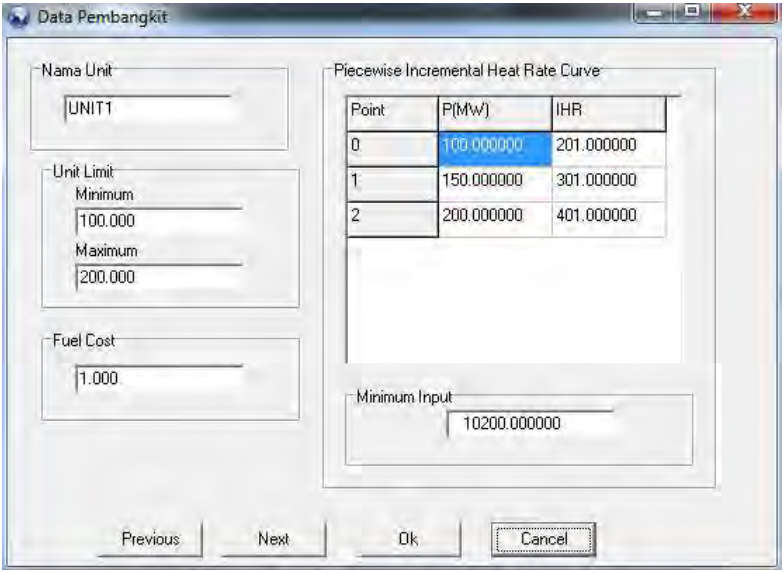
3.1.1 Menu EDC (*Economic Dispatch*)

Menu EDC ini adalah menu pada program Powergen yang digunakan untuk melakukan perhitungan *economic dispatch*. Program Powergen ini mampu melakukan perhitungan *economic dispatch* hingga 20 unit pembangkit dengan tiga jenis tipe kurva. Tampilan menu EDC dapat dilihat pada Gambar 3.2



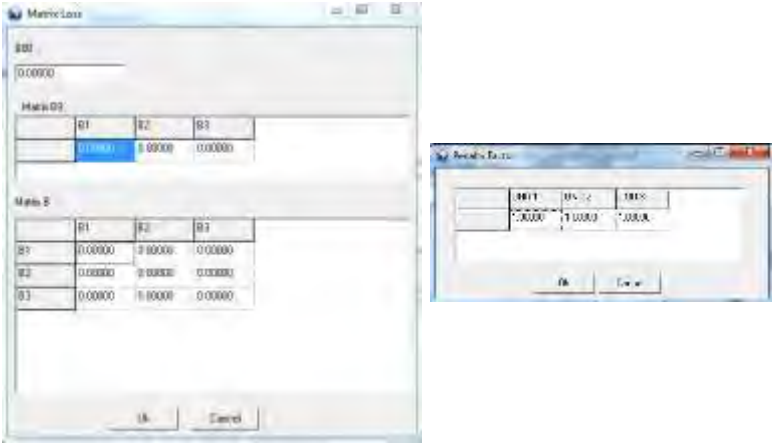
Gambar 3.2 Tampilan Utama Menu EDC

User/pengguna *software* mula mula mengisi data pembangkit dengan cara menekan tombol **Tambah** dan mengisi data-data pembangkit. *User* dapat mengisi kan data-data yang dimiliki oleh unit pembangkit seperti batasan pembangkitan minimum dan maksimum; karakteristik pembangkitan; serta biaya bahan bakar. Tampilan pengisian data-data pembangkit seperti pada Gambar 3.3.



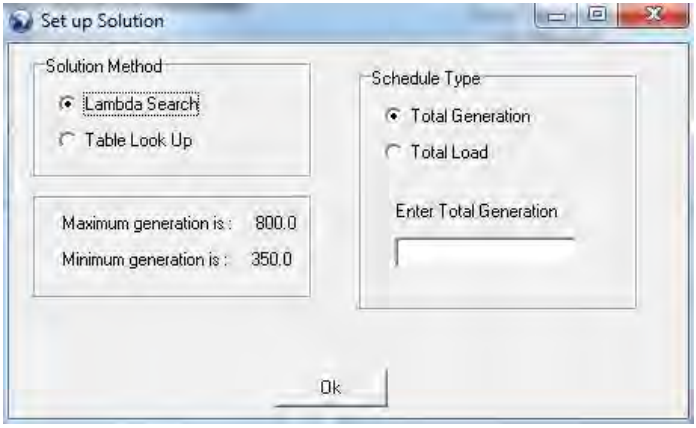
Gambar 3.3 Tampilan Pengisian Data Pembangkit

User pun dapat memilih metode perhitungan *losses* yang akan digunakan. Terdapat dua tipe perhitungan *losses* pada menu EDC ini jika *user* ingin memasukkan faktor *losses* kedalam perhitungan *economic dispatch*-nya. Tipe perhitungan *losses* yang pertama adalah dengan matriks B_{loss}. Tipe perhitungan *losses* yang kedua adalah dengan *constant penalty factor*. Namun dalam tugas akhir ini *losses* yang ada akan diabaikan. Tampilan pengisian *losses* dapat dilihat pada Gambar 3.4.



.**Gambar 3.4** Tampilan pilihan pengisian metode perhitungan *losses*

Setelah mengisikan data-data yang dibutuhkan *user* dapat melanjukan ke proses selanjutnya dengan memilih tombol **Run**. Ketika user memilih tombol **Run**, maka akan muncul tampilan seperti Gambar 3.5.



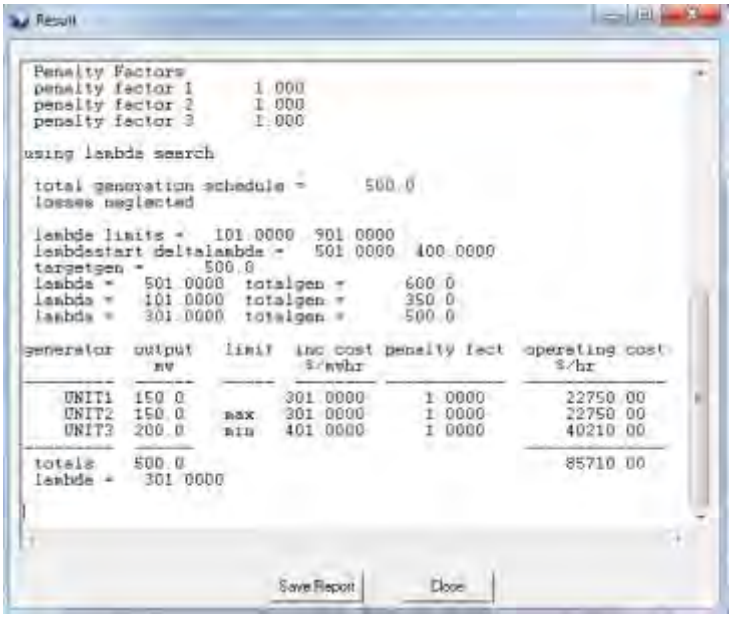
Gambar 3.5 Tampilan *Set up Solution* menu EDC

Pada tampilan ini *user* dapat memilihkan metode optimasi yang akan digunakan. Metode optimasi **Lambda Search** dikhususkan untuk

melakukan perhitungan dengan tipe kurva *polynomial* dan *piecewise incremental heat rate*. Metode optimasi **Table Look up** dikhususkan untuk melakukan perhitungan dengan tipe kurva *piecewise input/output*. Pada tugas akhir ini yang akan dikembangkan hanya pada metode **Lambda Search** dan tipe kurva *polynomial* dan *piecewise incremental heat rate*.

Selain itu pada tampilan ini *user* dapat melihat pembangkikatn maksimum dan pembangkitan minimum yang dapat di lakukan oleh unit-unit pembangkit yang ada. Selanjutnya *user* mengisikan **Total Generation** atau **Total Load** yang diinginkan sesuai **Schedule Type** yang dipilih.

Ketika semua data telah diisikan *user* dapat memilih **Ok** untuk menampilkan hasil perhitungan optimasi seperti Gambar 3.6.

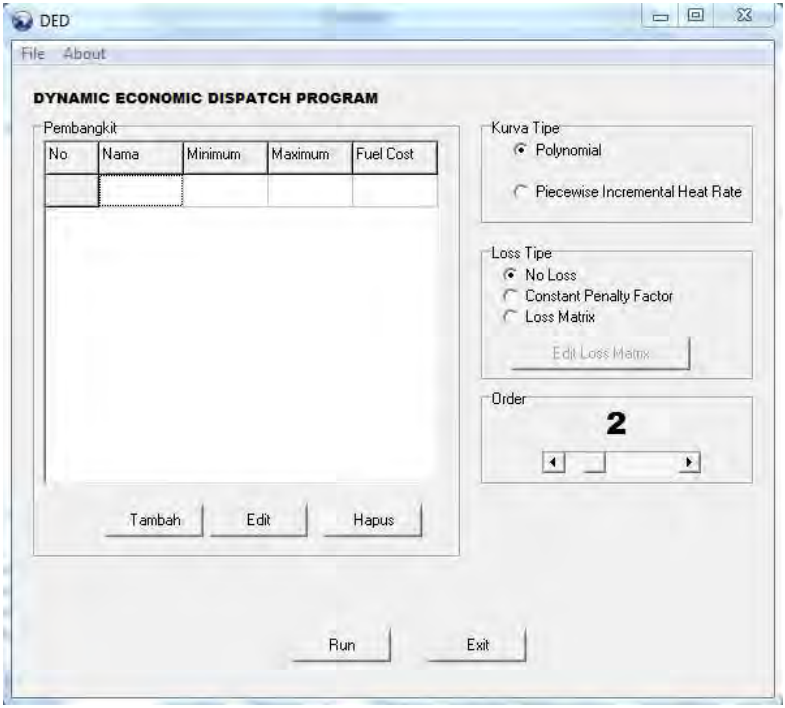


Gambar 3.6 Tampilan hasil perhitungan EDC

Pada tampilan hasil, *user* dapat memilih untuk menyimpan hasil perhitungan dengan menekan tombol **Save Report** atau dapat langsung menutup tampilan hasil dengan menekan tombol **Close**.

3.1.2 Menu DED (*Dynamic Economic Dispatch*)

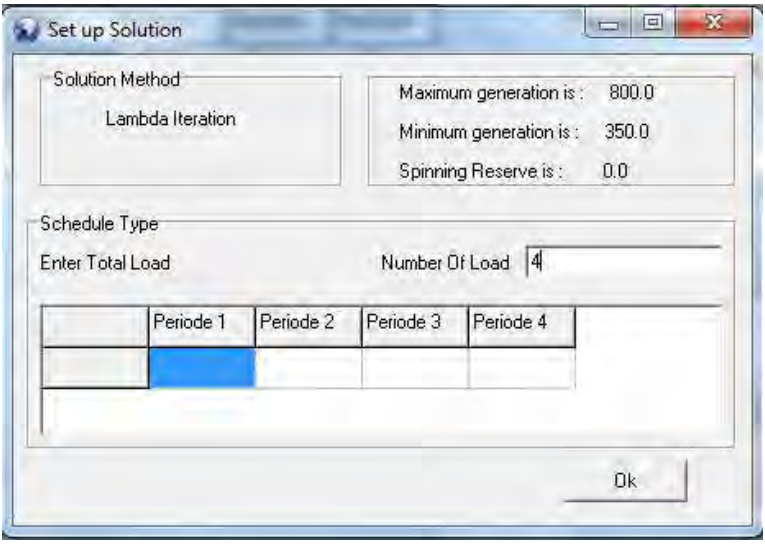
Menu DED ini adalah menu pada program Powergen yang digunakan untuk melakukan perhitungan *dynamic economic dispatch*. Secara umum menu ini hampir sama dengan menu EDC. Menu ini dapat melakukan perhitungan DED dengan jumlah unit mencapat 20 unit dengan dua tipe kurva. Hal ini disebabkan pada menu ini metode optimasi yang dapat digunakan adalah metode optimasi iterasi lambda. . Tampilan menu DED dapat dilihat pada Gambar 3.7.



Gambar 3.7 Tampilan menu DED

Dari segi tatap muka, baik menu EDC dan DED sekilas terlihat sama. Tampilan pengisian data pembangkit dan *losses* nya sama seperti pada menu EDC (Gambar 3.3 dan Gambar 3.4). Terdapat perbedaan

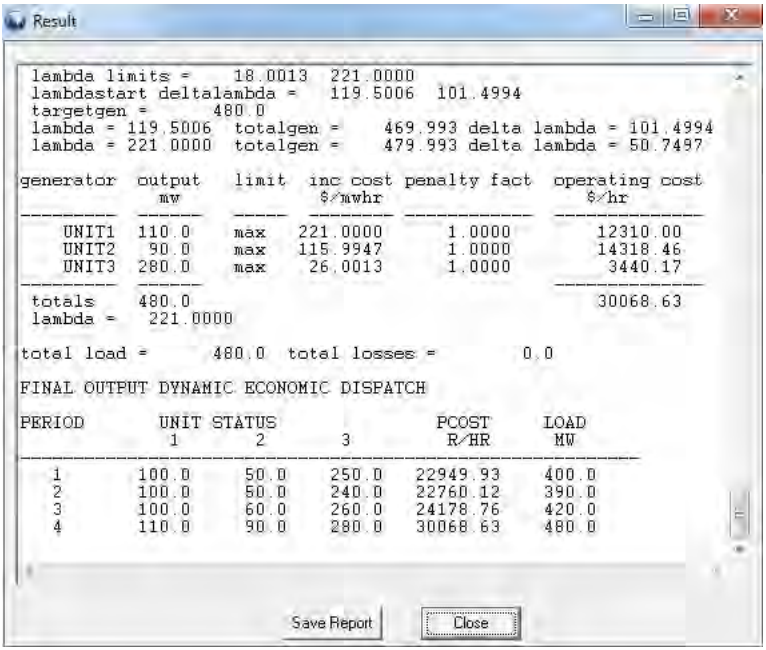
pada tampilan *Set up Solution* pada menu DED. Selain itu Untuk melihat perbedaannya dapat diperhatikan pada Gambar 3.8.



Gambar 3.8 Tampilan *Set up Solution* menu DED

Pada menu DED hanya tersedia metode penyelesaian iterasi lambda untuk dipilih oleh *user*. Selain itu perbedaan yang sangat jelas terlihat pada tampilan ini adalah pengisian jumlah beban. Pada menu DED ini *user* dapat mengisikan jumlah periode pembebanan pada kolom **Number of Load** untuk menampilkan sejumlah periode yang diinginkan pada kolom di bawahnya.

Ketika semua data telah diisikan *user* dapat menekan **Ok** untuk menampilkan hasil perhitungan optimasi seperti Gambar 3.9.



Gambar 3.9 Tampilan hasil perhitungan DED

Pada tampilan hasil menu DED, terdapat rekapitulasi dari setiap periode pembebanan. *User* dapat memilih untuk menyimpan hasil perhitungan dengan menekan tombol **Save Report** atau dapat langsung menutup tampilan hasil dengan menekan tombol **Close**.

3.2 Kasus Pengujian

Kasus pengujian digunakan untuk menguji kebenaran perhitungan yang dilakukan oleh *software*. Dalam tugas akhir ini akan ada lima kasus pengujian yang terbagi dalam dua bagian. Kasus 1 dan kasus 2 merupakan kasus yang digunakan untuk pengujian awal *software*. Pengujian awal bertujuan untuk menguji kelancaran penggunaan program. Sedangkan kasus 3, 4, dan 5 merupakan kasus yang digunakan untuk pengujian akhir *software.* Pengujian akhir bertujuan untuk menguji ketepatan perhitungan yang dilakukan oleh program sebelum

dan setelah mempertimbangkan batasan *ramp-rate* dan *spinning reserve*.

Pada kasus 1 dan kasus 2, kedua data pengujian diambil dari soal pada buku "*Power System Generation, Operation, and Control*"[7]. Keduanya pun memiliki dua hingga tiga buah unit pembangkit dengan dua pola beban. Karakteristik pembangkit pada kasus pertama adalah kurva polynomial orde dua, sedangkan pada kasus kedua adalah *piecewise incremental heat*.

Pada kasus 3, 4, dan 5, data pengujian diambil dari paper referensi[1]. Data pembangkit yang digunakan pada ketiga kasus tersebut sama, yang berbeda adalah pada batasan yang dipertimbangkan. Kasus 3 berupa pengujian tanpa mempertimbangkan *ramp-rate* dan *spinning reserve*. Kasus 4 berupa pengujian dengan mempertimbangkan *ramp-rate* saja. Kasus 5 berupa pengujian dengan mempertimbangkan *ramp-rate* dan *spinning reserve*.

Seperti yang sudah dinyatakan sebelumnya, kelima kasus pengujian ini akan mengabaikan perhitungan rugi jaringan (*losses*) yang ada.

3.2.1 Kasus 1

Berikut adalah data pengujian awal kasus pertama. Kasus ini menggunakan tipe kurva polynomial orde dua dengan *individual spinning reserve*

$$H_1 = 225 + 8.4P_1 + 0.0025P_1^2$$
$$H_2 = 729 + 6.3P_2 + 0.0081P_2^2$$
$$H_3 = 400 + 7.5P_3 + 0.0025P_3^2$$

Tabel 3.1 Data kasus 1

Unit	Minimum (MW)	Maksimum (MW)	Fuel Cost ($/Mbtu)	DR/UR (MW/h)	% Spinning Reserve
1	45	350	0.8	50	10%
2	45	350	1.02	100	15%
3	47.5	450	0.9	50	20%

Pada periode sebelumnya sudah ada pembebanan sebagai berikut
$P_1 = 276.3\ MW$ $P_2 = 84.7\ MW$ $P_3 = 239.0$ MW
Beban pada periode ini $P_L = 750\ MW$

3.2.2 Kasus 2

Berikut adalah data pengujian awal kasus kedua. Kasus ini menggunakan tipe kurva *piecewise incremental heat-rate* dengan *spinning reserve* yang sama.

Spinning reserve sistem : 20% Total kapasitas pembangkit

Unit 1: $P_1^{min} = 100MW \;\; P_1^{max} = 400MW \;\; DR = UR = \;\; 50MW/h$

Unit 2: $P_2^{min} = 120MW \;\; P_2^{max} = 300MW \;\; DR = UR = \;\; 80MW/h$

Unit 3: $P_3^{min} = 150\ MW \;\; P_3^{max} = 450MW \;\; DR = UR = \;\; 70MW/h$

Tabel 3.2 Data kasus 2

$\boldsymbol{P_1(MW)}$	$\boldsymbol{Incremental\ Cost}$ $(\$/\boldsymbol{MWh})$
100	6.5
200	7.0
300	8.0
400	11.0
No load cost	300 *MW*
$\boldsymbol{P_2(MW)}$	$\boldsymbol{Incremental\ Cost}$ $(\$/\boldsymbol{MWh})$
120	8.0
150	8.3
200	9.0
300	12.5
No load cost	425 *MW*
$\boldsymbol{P_3(MW)}$	$\boldsymbol{Incremental\ Cost}$ $(\$/\boldsymbol{MWh})$
150	7.5
275	7.7
390	8.1
450	8.5
No load cost	400 *MW*

Pada periode sebelumnya sudah ada pembebanan sebagai berikut

$P_1 = 279\ MW \quad P_2 = 120\ MW \quad P_3 = 301\ MW$

Beban pada periode ini $P_L = 850\ MW$

3.2.3 Kasus 3

Mulai dari kasus 3 hingga kasus 5 akan menggunakan data unit pembangkit yang sama. Rincian data pembangkit yang akan digunakan pada ketiga kasus selanjutnya adalah sebagai berikut.

Tabel 3.3 Data Pembangkit untuk kasus 3, 4, dan 5

Unit	P^{min}	P^{max}	a	b	c
1	7	15.00	0.602842	22.45526	85.74158
2	7	45.00	0.602842	22.45526	85.74158
3	13	25.00	0.214263	22.52789	108.9837
4	16	25.00	0.077837	26.75263	49.06263
5	16	25.00	0.077837	26.75263	49.06263
6	3	14.75	0.734763	80.39345	677.73000
7	3	14.75	0.734763	80.39345	677.73000
8	3	12.28	0.514474	13.19474	44.39000
9	3	12.28	0.514474	13.19474	44.39000
10	3	12.28	0.514474	13.19474	44.39000
11	3	12.28	0.514474	13.19474	44.39000
12	3	24.00	0.657079	56.70947	574.96030
13	3	16.20	1.236474	84.67579	820.37760
14	3	36.20	0.394571	59.59026	603.02370
15	3	45.00	0.420789	56.70947	567.93630
16	3	37.00	0.420789	55.96500	567.93630
17	3	45.00	0.420789	55.96500	567.93630
18	3	16.20	1.236474	84.67579	820.37760

Seluruh unit pembangkit tersebut akan di hitung pembebanan optimalnya untuk empat periode pembebanan sebagai berikut

Tabel 3.4 Total Pembangkitan untuk kasus 3, 4, dan 5

Periode ke-	Total Pembangkitan (MW)
1	411.559
2	389.898
3	346.576
4	303.254

Pada kasus 3 batasan *ramp-rate* dan *spinning reserve* tidak diperhitungkan.

3.2.4 Kasus 4

Pada kasus 4 data pembangkit serta total pembangkitan diambil dari kasus 3 (Tabel 3.3 dan Tabel 3.4). Perbedaan dari kasus sebelumnya adalah pada kasus ini akan diperhitungkan batasan *ramp-rate*. Data batasan *ramp-rate* untuk tiap pembangkit dapat dilihat pada Tabel 3.5.

Tabel 3.5 Data *ramp-rate* unit pembangkit

Unit	DR (MW/jam)	UR (MW/jam)
1	10	10
2	10	10
3	10	10
4	10	10
5	10	10
6	5	5
7	10	10
8	10	10
9	10	10
10	10	10
11	10	10
12	10	10
13	10	10
14	7	7
15	10	10
16	10	10
17	10	10
18	3	3

3.2.5 Kasus 5

Pada kasus 5 batasan *ramp rate* dan *spinning reserve* akan diperhitungkan. Syarat spinning reserve dari sistem seperti pada Tabel 3.6

Tabel 3.6 Data *spinning reserve* unit pembangkit

Unit	P^{max}	Spinning Reserve Minimum	
		(%)	(MW)
1	15.00	0	0
2	45.00	5	2.25
3	25.00	12	3
4	25.00	12	3
5	25.00	12	3

Unit	P^{max}	Spinning Reserve Minimum	
		(%)	(MW)
6	14.75	0	0
7	14.75	0	0
8	12.28	10	1.228
9	12.28	10	1.228
10	12.28	10	1.228
11	12.28	10	1.228
12	24.00	0	0
13	16.20	0	0
14	36.20	0	0
15	45.00	5	2.25
16	37.00	0	0
17	45.00	5	2.25
18	16.20	0	0
Jumlah			20.662

3.3 Perhitungan Manual

Sub bab ini menjelaskan tentang pergitungan manual untuk setiap kasus pengujian yang ada. Pada kasus 1 digunakan metode persamaan *lagrange* untuk menyederhanakan perhitungan manual. Pada kasus 2 digunakan metode iterasi lambda karena sifat kurva *piecewise* yang tidak memungkinkan menggunakan *lagrange*. Kasus 3 hingga 5 menggunakan persamaan *lagrange* yang disederhanakan.

3.3.1 Perhitungan Kasus 1

Data pembangkit untuk kasus 1 dapat dilihat pada Tabel 3.1. Hal pertama yang perlu diperhatikan adalah *spinning reserve* dan *ramp-rate*.

Tabel 3.7 *Spinning Reserve* kasus 1

Unit	Kapasitas (MW)	Spinning Reserve		P_i^{max} (MW)
		(%)	(MW)	
1	350	10	35.0	315.0
2	350	15	52.5	297.5
3	450	20	90.0	360.0

Tabel 3.8 *Ramp-rate* kasus 1

Unit	$P_{i,t-1}$ (MW)	DR & UR (MW)	$P_{i,t-1} - DR$ (MW)	$P_{i,t-1} + UR$ (MW)
1	276.3	50	226.3	326.3
2	84.7	100	0.0	184.7
3	239.0	50	189.0	289.0

$$P_i^{min} = \max\left(P_i^{min}, P_{i,t-1} - DR\right)$$
$$P_i^{max} = \min\left(P_i^{max}, P_{i,t-1} + UR\right)$$

Sehingga batasan pembangkitan unit yang baru menjadi

$P_1^{min} = 226.3\ MW$ $\quad$ $P_1^{max} = 315.0\ MW$
$P_2^{min} = 45.0\ MW$ $\quad$ $P_2^{max} = 184.7\ MW$
$P_3^{min} = 189.0\ MW$ $\quad$ $P_3^{max} = 289.0\ MW$

Dari data fungsi ihr, kita dapat menentukan fungsi biaya dengan mengalikan fungsi ihr dengan *fuel cost*

$$F_1 = 180 + 6.72P_1 + 0.00200P_1^2$$
$$F_2 = 744 + 6.43P_2 + 0.00826P_2^2$$
$$F_3 = 360 + 6.75P_3 + 0.00225P_3^2$$

Berdasarkan persamaan (2.2) dan (2.20) maka

$$\frac{dF_1}{dP_1} = 6.72 + 0.0040P_1$$
$$\frac{dF_2}{dP_2} = 6.43 + 0.0165P_1$$
$$\frac{dF_3}{dP_3} = 6.72 + 0.0045P_1$$

$$\lambda = \frac{P_L + \sum \frac{b_i}{2a_i}}{\sum \frac{1}{2a_i}}$$

$$\lambda = \frac{750 + \frac{6.72}{0.004} + \frac{6.43}{0.0165} + \frac{6.72}{0.0045}}{\frac{1}{0.004} + \frac{1}{0.0165} + \frac{1}{0.0045}} = 8.0946$$

$$P_i = \frac{\lambda - b_i}{2a_i}$$

$P_1 = 343.65\ MW$ (max)
$P_2 = 100.88\ MW$
$P_3 = 305.47\ MW$ (max)

Karena P_1 dan P_3 melebihi batas maksimum, maka kelebihannya diberikan seluruhnya pada P_2 sehingga

$P_1 = 315.0\ MW$
$P_2 = 146.0\ MW$
$P_3 = 289.0\ MW$

3.3.2 Perhitungan Kasus 2

Data pembangkit untuk kasus 1 dapat dilihat pada Tabel 3.1. Hal pertama yang perlu diperhatikan adalah *spinning reserve* dan *ramp-rate*.

Tabel 3.9 *Spinning reserve* kasus 2

Unit	Kapasitas (MW)	Spinning Reserve		P_i^{max} (MW)
		(%)	(MW)	
1	400	20	80.0	320.0
2	300	20	60.0	240.0
3	450	20	90.0	360.0

Tabel 3.10 *Ramp-rate* kasus 2

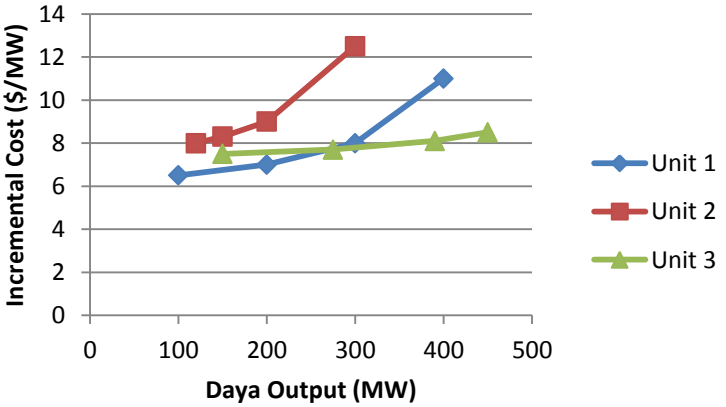
Unit	$P_{i,t-1}$ (MW)	DR & UR (MW)	$P_{i,t-1} - DR$ (MW)	$P_{i,t-1} + UR$ (MW)
1	279.0	50	229.0	329.0
2	120.0	80	40.0	200.0
3	301.0	70	231.0	371.0

$$P_i^{min} = \max\left(P_i^{min}, P_{i,t-1} - DR\right)$$
$$P_i^{max} = \min\left(P_i^{max}, P_{i,t-1} + UR\right)$$

Sehingga batasan pembangkitan unit yang baru menjadi

$P_1^{min} = 229.0\ MW$ $P_1^{max} = 320.0\ MW$
$P_2^{min} = 120.0\ MW$ $P_2^{max} = 200.0\ MW$
$P_3^{min} = 231.0\ MW$ $P_3^{max} = 360.0\ MW$

Dari data pada Tabel 3.1 dapat kita buat grafik *incremental cost* nya seperti pada Gambar 3.10



Gambar 3.10 Plot kurva *incremental cost* kasus 2

Dengan menggunakan persamaan (2.6) dan persamaan (2.7) kita bisa mendapatkan bentuk polynomial dari fungsi *piecewise* unit pembangkit.

$$\frac{dF_i}{dP_i} = \frac{y_2 - y_1}{x_2 - x_1} P_i + \frac{x_2 y_1 - x_1 y_2}{x_2 - x_1} \qquad , x_1 \le P_i \le x_2$$

Fungsi *incremental cost* :

$$\frac{dF_1}{dP_1} = \begin{cases} 0.005\ P_1 + 6.0 & , 100 \le P_1 \le 200 \\ 0.010\ P_1 + 5.0 & , 200 \le P_1 \le 300 \\ 0.030\ P_1 - 1.0 & , 300 \le P_1 \le 400 \end{cases}$$

$$\frac{dF_2}{dP_2} = \begin{cases} 0.010\ P_2 + 6.8 & , 100 \le P_1 \le 200 \\ 0.014\ P_2 + 6.2 & , 200 \le P_1 \le 300 \\ 0.035\ P_2 + 2.0 & , 300 \le P_1 \le 400 \end{cases}$$

$$\frac{dF_3}{dP_3} = \begin{cases} 0.0016\ P_3 + 7.26 & , 100 \le P_1 \le 200 \\ 0.0035\ P_3 + 6.74 & , 200 \le P_1 \le 300 \\ 0.0067\ P_3 + 5.50 & , 300 \le P_1 \le 400 \end{cases}$$

Fungsi biaya :

$$F_1 = \begin{cases} 0.0025\ P_1^2 + 6.0\ P_1 + 300 & ,100 \le P_1 \le 200 \\ 0.0050\ P_1^2 + 5.0\ P_1 + 300 & ,200 \le P_1 \le 300 \\ 0.0150\ P_1^2 - 1.0\ P_1 + 300 & ,300 \le P_1 \le 400 \end{cases}$$

$$F_2 = \begin{cases} 0.0050\ P_2^2 + 6.8\ P_2 + 425 & ,100 \le P_1 \le 200 \\ 0.0070\ P_2^2 + 6.2\ P_2 + 425 & ,200 \le P_1 \le 300 \\ 0.0175\ P_2^2 + 2.0\ P_2 + 425 & ,300 \le P_1 \le 400 \end{cases}$$

$$F_3 = \begin{cases} 0.00080\ P_3 + 7.26\ P_3 + 400 & ,100 \le P_1 \le 200 \\ 0.00175\ P_3 + 6.74\ P_3 + 400 & ,200 \le P_1 \le 300 \\ 0.00335\ P_3 + 5.50\ P_3 + 400 & ,300 \le P_1 \le 400 \end{cases}$$

Kasus ini cukup rumit bila kita selesaikan dengan persamaan *lagrange*, oleh karena itu akan digunakan metode iterasi lambda.

Perhitungan untuk $P_L = 850\ MW$

Menentukan lambda max dan lambda min sebagai referensi awal iterasi lambda

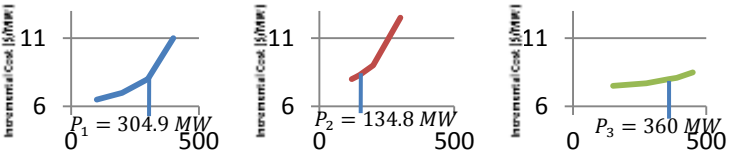
$\lambda_{max} = \max[\lambda_1(P_1^{max}),\ \lambda_2(P_2^{max}),\ \lambda_3(P_3^{max})]$
$\lambda_{min} = \min[\lambda_1(P_1^{min}),\ \lambda_2(P_2^{min}),\ \lambda_3(P_3^{min})]$

Didapatkan $\lambda_{max} = 7.2900$ $\lambda_{min} = 9.0000$

Untuk memulai iterasi lambda diambil nilai tengah dari λ_{max} dan λ_{min}

$$\lambda_{start} = \frac{\lambda_{max} + \lambda_{min}}{2} = 8.145$$

Untuk $\lambda = 8.145$, setiap unit akan memiliki pembangkitan seperti yang terlihat pada Gambar 3.11



Gambar 3.11 Ilustrasi kasus 2 iterasi pertama

$\sum P_i = 799.71\ MW$

Karena $\sum P_i < Target\ gen$ maka lambda harus ditambah sebesar $\Delta\lambda$ dimana

$$\Delta\lambda = \frac{\lambda_{max} - \lambda_{min}}{2} = 1.9500$$

Pada iterasi ini $\lambda = 9.0000$, $\sum P_i = 880.00\ MW$

Agar konvergen, maka delta lambda pada iterasi ini menjadi setengah dari lambda sebelumnya

$$\Delta\lambda = 0.9750$$

Begitu selanjutnya hingga didapatkan $\sum P_i = 850\ MW$

$\lambda =$	8.145	$\sum P_i =$	799.33	MW	$\Delta\lambda =$	0.855
$\lambda =$	9.000	$\sum P_i =$	880.00	MW	$\Delta\lambda =$	0.4275
$\lambda =$	8.5725	$\sum P_i =$	848.55	MW	$\Delta\lambda =$	0.2138
$\lambda =$	8.7862	$\sum P_i =$	864.73	MW	$\Delta\lambda =$	0.1069
$\lambda =$	8.6794	$\sum P_i =$	857.09	MW	$\Delta\lambda =$	0.0534
$\lambda =$	8.6259	$\sum P_i =$	853.28	MW	$\Delta\lambda =$	0.0267
$\lambda =$	8.5992	$\sum P_i =$	851.35	MW	$\Delta\lambda =$	0.0134
$\lambda =$	8.5859	$\sum P_i =$	849.95	MW	$\Delta\lambda =$	0.0067
$\lambda =$	8.5925	$\sum P_i =$	850.65	MW	$\Delta\lambda =$	0.0033
$\lambda =$	8.5892	$\sum P_i =$	850.29	MW	$\Delta\lambda =$	0.0017
$\lambda =$	8.5875	$\sum P_i =$	850.12	MW	$\Delta\lambda =$	0.0008
$\lambda =$	8.5867	$\sum P_i =$	850.04	MW	$\Delta\lambda =$	0.0004
$\lambda =$	8.5863	$\sum P_i =$	849.99	MW		

Pada iterasi terakhir sudah didapatkan nilai pembangkitan yang mendekati nilai target pembangkitan dengan perincian

$P_1 = 319.5\ MW$

$P_2 = 170.5\ MW$

$$P_3 = 360\ MW$$

3.3.3 Perhitungan Kasus 3

Data pembangkit untuk kasus 3 dapat dilihat pada tabel (3.3). Dari persamaan (2.2) maka untuk setiap unit pembangkit didapatkan persamaan *incremental cost* sebagai berikut.

Tabel 3.11 *Incremental cost function* kasus 3,4, dan 5

Unit	*Incremental Cost Function*	$1/2a$	*b/2a*
1	1.205684 P + 22.45526	0.8294047	18.624499
2	1.205684 P + 22.45526	0.8294047	18.624499
3	0.428526 P + 22.52789	2.3335807	52.570649
4	0.155674 P + 26.75263	6.4236803	171.85034
5	0.155674 P + 26.75263	6.4236803	171.85034
6	1.469526 P + 80.39345	0.6804915	54.707062
7	1.469526 P + 80.39345	0.6804915	54.707062
8	1.028948 P + 13.19474	0.9718664	12.823525
9	1.028948 P + 13.19474	0.9718664	12.823525
10	1.028948 P + 13.19474	0.9718664	12.823525
11	1.028948 P + 13.19474	0.9718664	12.823525
12	1.314158 P + 56.70947	0.7609435	43.152703
13	2.472948 P + 84.67579	0.4043757	34.240829
14	0.789142 P + 59.59026	1.2671991	75.512721
15	0.841578 P + 56.70947	1.1882440	67.384687
16	0.841578 P + 55.96500	1.1882440	66.500075
17	0.841578 P + 55.96500	1.1882440	66.500075
18	2.472948 P + 84.67579	0.4043757	34.240829
	Jumlah	28.48982522	981.7604705

Pembebanan pertama

$$P_{L,1} = 411.559\ MW$$

Berdasarkan tabel diatas kita dapat menghitung lambda dengan menggunakan persamaan (2.20)

$$\lambda = \frac{P_L + \sum \frac{b_i}{2a_i}}{\sum \frac{1}{2a_i}}$$

$$\lambda = \frac{411.559 + 981.76}{28.49} = 48.905$$

Dengan lambda tersebut kita dapat menghitung pembangkitan yang dilakukan oleh tiap unit dengan persamaan (2.18). pembangkitan oleh tiap unit pada iterasi ini dapat dilihat pada tabel berikut.

Tabel 3.12 Perhitungan manual kasus 3 periode pertama iterasi 1

Unit	P_i^{min}	P_i^{max}	P_i	status
1	7	15	21.30765	max
2	7	45	21.30765	
3	13	25	59.78089	max
4	16	25	137.4213	max
5	16	25	137.4213	max
6	3	14.75	-21.9444	min
7	3	14.75	-21.9444	min
8	3	12.28	33.96752	max
9	3	12.28	33.96752	max
10	3	12.28	33.96752	max
11	3	12.28	33.96752	max
12	3	24	-6.51665	min
13	3	16.2	-14.7719	Min
14	3	36.2	-14.5027	Min
15	3	45	-10.176	Min
16	3	37	-9.29141	Min

Unit	P_i^{min}	P_i^{max}	P_i	status
17	3	45	-9.29141	Min
18	3	16.2	-14.7719	Min

Karena ada unit-unit yang melanggar batasan daya pembangkitan (Unit 1,3,4,5,8,9,10,11), maka unit-unit yang melampaui batasan maksimumnya kita maksimum kan dan kita hitung berapa daya sisa nya yang harus kita optimumkan.

$P_{L,1}^{(1)} = P_L - \sum P_j^{max} = 272.439\ MW$

Unit 2,6,7,12,13,14,15,16,17, dan 18 akan kita hitung kembali lambda nya

$$\lambda^{(1)} = \frac{272.439 + 515.571}{8.592} = 91.714$$

Tabel 3.13 Perhitungan manual kasus 3 periode pertama iterasi 2

Unit	P_i^{min}	P_i^{max}	P	Status
2	7	45	57.44368	max
6	3	14.75	7.70366	
7	3	14.75	7.70366	
12	3	24	26.636612	max
13	3	16.2	2.84615	min
14	3	36.2	40.707409	max
15	3	45	41.594144	
16	3	37	42.478756	max
17	3	45	42.478756	
18	3	16.2	2.846156	min

Iterasi selanjutnya

$P_{L,1}^{(2)} = P_L - \sum P_j^{max} = 130.239\ MW$

$$\lambda^{(2)} = \frac{130.239 + 311.780}{4.546} = 97.228$$

Tabel 3.14 Perhitungan manual kasus 3 periode pertama iterasi 3

Unit	P_i^{min}	P_i^{max}	P	status
6	3	14.75	11.4556840	
7	3	14.75	11.4556840	
13	3	16.2	5.0757580	
15	3	45	48.1457519	max
17	3	45	49.0303639	max
18	3	16.2	5.0757580	

Iterasi ke tiga

$$P_{L,1}^{(3)} = P_L - \sum P_j^{max} = 40.239\ MW$$

$$\lambda^{(3)} = \frac{40.239 + 177.896}{2.168} = 100.535$$

Tabel 3.15 Perhitungan manual kasus 3 periode pertama iterasi 4

Unit	P_i^{min}	P_i^{max}	P	Status
6	3	14.75	13.7063217	
7	3	14.75	13.7063217	
13	3	16.2	6.413178	
18	3	16.2	6.413178	

Berikut adalah hasil akhir *economic dispatch* untuk beban pertama setelah 4 iterasi

Tabel 3.16 Hasil kasus 3 periode pertama

UNIT	$P_{i,1}$	status	λ	Prod Cost
UNIT1	15	max	40.5405	558.21
UNIT2	45	max	76.711	2316.98
UNIT3	25	max	33.241	806.10

UNIT	$P_{i,1}$	status	λ	Prod Cost
UNIT4	25	max	30.6445	766.53
UNIT5	25	max	30.6445	766.53
UNIT6	13.7		100.5537	1918.92
UNIT7	13.7		100.5537	1918.92
UNIT8	12.3	max	25.8302	284.00
UNIT9	12.3	max	25.8302	284.00
UNIT10	12.3	max	25.8302	284.00
UNIT11	12.3	max	25.8302	284.00
UNIT12	24	max	88.2493	2314.47
UNIT13	6.4		100.5537	1415.02
UNIT14	36.2	max	88.1572	3277.25
UNIT15	45	max	94.5805	3971.96
UNIT16	37	max	87.1034	3214.70
UNIT17	45	max	93.836	3938.46
UNIT18	6.4		100.5537	1415.02

Untuk pembebanan ke-2 dan selanjutnya, dilakukan dengan proses yang sama. Hasil akhir perhitungan manual untuk kasus 3 adalah sebagai berikut.

Tabel 3.17 Hasil akhir kasus 3

Unit	Time Interval			
	1	2	3	4
1	15.0	15.0	15.0	15.0
2	45.0	45.0	45.0	44.6
3	25.0	25.0	25.0	25.0
4	25.0	25.0	25.0	25.0
5	25.0	25.0	25.0	25.0
6	13.7	8.2	3.0	3.0

Unit	Time Interval			
	1	2	3	4
7	13.7	8.2	3.0	3.0
8	12.3	12.3	12.3	12.3
9	12.3	12.3	12.3	12.3
10	12.3	12.3	12.3	12.3
11	12.3	12.3	12.3	12.3
12	24.0	24.0	20.7	14.9
13	6.4	3.1	3.0	3.0
14	36.2	36.2	30.9	21.1
15	45.0	42.5	32.4	23.2
16	37.0	37.0	33.2	24.1
17	45.0	43.4	33.2	24.1
18	6.4	3.1	3.0	3.0
Biaya	29731.07	27653.74	23855.28	20386.21
Beban	411.6	389.9	346.6	303.3

Pada data tabel terlihat pembangkitan untuk seluruh pembangkit untuk setiap periode pembebanan. Total biaya untuk keempat periode tersebut didapatkan sejumlah $ 101626.3 .

3.3.4 Perhitungan Kasus 4

Karena pada kasus 4 ini mempertimbangkan *ramp-rate*, maka untuk periode ke 2 dan seterusnya akan berbeda. Untuk hasil perhitungan pertama dapat dilihat pada Tabel 3.13

Dengan adanya batasan *ramp-rate* maka P^{min} dan P^{max} berubah menjadi seperti berikut

Tabel 3.18 *Ramp-rate* kasus 4 periode 2

Unit	$P_{i,t-1}$ (MW)	DR & UR (MW)	$P_{i,t-1} - DR$ (MW)	$P_{i,t-1} + UR$ (MW)	Setelah Ramp-rate	
					P_i^{min}	P_i^{max}
1	15.0	10	5.0	25.0	7.0	15.0
2	45.0	10	35.0	55.0	35.0	45.0
3	25.0	10	15.0	35.0	15.0	25.0

Unit	$P_{i,t-1}$ (MW)	DR & UR (MW)	$P_{i,t-1} - DR$ (MW)	$P_{i,t-1} + UR$ (MW)	Setelah Ramp-rate	
					P_i^{min}	P_i^{max}
4	25.0	10	15.0	35.0	16.0	25.0
5	25.0	10	15.0	35.0	16.0	25.0
6	13.7	5	8.7	18.7	8.7	14.8
7	13.7	10	3.7	23.7	3.7	14.8
8	12.3	10	2.3	22.3	3.0	12.3
9	12.3	10	2.3	22.3	3.0	12.3
10	12.3	10	2.3	22.3	3.0	12.3
11	12.3	10	2.3	22.3	3.0	12.3
12	24.0	10	14.0	34.0	14.0	24.0
13	6.4	10	0	16.4	3.0	16.2
14	36.2	7	29.2	43.2	29.2	36.2
15	45.0	10	35.0	55.0	35.0	45.0
16	37.0	10	27.0	47.0	27.0	37.0
17	45.0	10	35.0	55.0	35.0	45.0
18	6.4	3	3.4	9.4	3.4	9.4

Oleh karena itu untuk periode kedua kasus 4 hasil perhitungannya sebagai berikut.

Tabel 3.19 Hasil kasus 4 periode kedua

UNIT	$P_{i,2}$	Status	λ	Biaya
UNIT1	15	max	40.5405	558.21
UNIT2	45	max	76.711	2316.98
UNIT3	25	max	33.241	806.1
UNIT4	25	max	30.6445	766.53
UNIT5	25	max	30.6445	766.53
UNIT6	8.7	min	93.1876	1433.36
UNIT7	8.1		92.2445	1373.86
UNIT8	12.3	max	25.8302	284
UNIT9	12.3	max	25.8302	284
UNIT10	12.3	max	25.8302	284
UNIT11	12.3	max	25.8302	284
UNIT12	24	max	88.2493	2314.47
UNIT13	3.1		92.2445	1091.12

UNIT	$P_{i,2}$	Status	λ	Biaya
UNIT14	36.2	max	88.1572	3277.25
UNIT15	42.2		92.2445	3712.68
UNIT16	37	max	87.1034	3214.7
UNIT17	43.1		92.2445	3762.51
UNIT18	3.4	min	93.1164	1123.8

Hal tersebut dilakukan terus hingga periode ke empat. Hasil akhir untuk perhitungan manual kasus 4 tertera pada Tabel 3.17

Tabel 3.20 Hasil akhir kasus 4

Unit	Time Interval			
	1	2	3	4
1	15.0	15.0	15.0	15.0
2	45.0	45.0	45.0	44.2
3	25.0	25.0	25.0	25.0
4	25.0	25.0	25.0	25.0
5	25.0	25.0	25.0	25.0
6	13.7	8.7	3.7	3.0
7	13.7	8.1	3.0	3.0
8	12.3	12.3	12.3	12.3
9	12.3	12.3	12.3	12.3
10	12.3	12.3	12.3	12.3
11	12.3	12.3	12.3	12.3
12	24.0	24.0	20.6	14.5
13	6.4	3.1	3.0	3.0
14	36.2	36.2	30.7	23.7
15	45.0	42.2	32.2	22.6
16	37.0	37.0	33.1	23.5
17	45.0	43.1	33.1	23.5

Unit	Time Interval			
	1	2	3	4
18	6.4	3.4	3.0	3.0
Biaya	29731.07	27654.1	23856.3	20389.45
Beban	411.6	389.9	346.6	303.3

Pada data tabel terlihat pembangkitan untuk seluruh pembangkit untuk setiap periode pembebanan. Total biaya untuk keempat periode tersebut didapatkan sejumlah $ 101630.92.

3.3.5 Perhitungan Kasus 5

Karena pada kasus 5 ini mempertimbangkan *spinning reserve*, maka pembangkitan maksimum dari setiap unit pembangkit akan dibatasi sesuai Tabel 3.18

Tabel 3.21 *Spinning reserve* kasus 5

Unit	**Kapasitas**	**Spinning Reserve Minimum**		P_i^{max}
		(%)	(MW)	
1	15.00	0	0	12.00
2	45.00	5	2.25	42.75
3	25.00	12	3	22.00
4	25.00	12	3	22.00
5	25.00	12	3	22.00
6	14.75	0	0	14.75
7	14.75	0	0	14.75
8	12.28	10	1.228	11.05
9	12.28	10	1.228	11.05
10	12.28	10	1.228	11.05
11	12.28	10	1.228	11.05
12	24.00	0	0	24.00
13	16.20	0	0	16.20
14	36.20	0	0	36.20
15	45.00	5	2.25	42.75
16	37.00	0	0	37.00
17	45.00	5	2.25	42.75
18	16.20	0	0	16.20
Jumlah			20.662	

Perhitungan selanjutnya akan sama seperti kasus kasus sebelumnya. Hasil akhir dari perhitungan manual kasus 5 adalah sebagai berikut

Tabel 3.22 Hasil akhir kasus 5

Unit	Time Interval			
	1	2	3	4
1	15.0	15.0	15.0	15.0
2	42.8	42.8	42.8	42.8
3	22.0	22.0	22.0	22.0
4	22.0	22.0	22.0	22.0
5	22.0	22.0	22.0	22.0
6	14.8	10.4	5.4	3.0
7	14.8	10.4	3.2	3.0
8	11.1	11.1	11.1	11.1
9	11.1	11.1	11.1	11.1
10	11.1	11.1	11.1	11.1
11	11.1	11.1	11.1	11.1
12	24.0	24.0	21.7	16.3
13	15.7	5.7	3.0	3.0
14	36.2	36.2	32.4	25.4
15	42.8	42.8	33.8	25.4
16	37.0	37.0	34.7	26.3
17	42.8	42.8	34.7	26.3
18	15.7	12.7	9.7	6.7
Biaya	31029.97	28702.75	24759.32	21135.12
Beban	411.6	389.9	346.6	303.3

Pada data tabel terlihat pembangkitan untuk seluruh pembangkit untuk setiap periode pembebanan. Total biaya untuk keempat periode tersebut didapatkan sejumlah $ 105627.16.

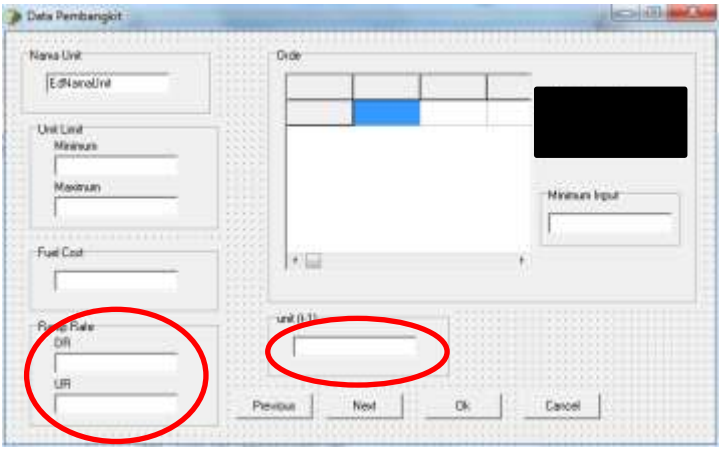
BAB 4
IMPLEMENTASI DAN PENGUJIAN *SOFTWARE*

4.1 Implementasi

Sub-bab ini menjelaskan tentang pengembangan program yang telah diimplementasikan batasan *ramp-rate* dan *spinning reserve*. Sub-bab ini akan lebih membahas tentang pengembangan dari sisi visualnya saja (*user interface*) dan juga sedikit membahas penambahan-penambahan yang dilakukan dari sisi kode logikanya (*logic code*). Untuk kode logika selengkapnya akan dicantumkan pada lampiran.

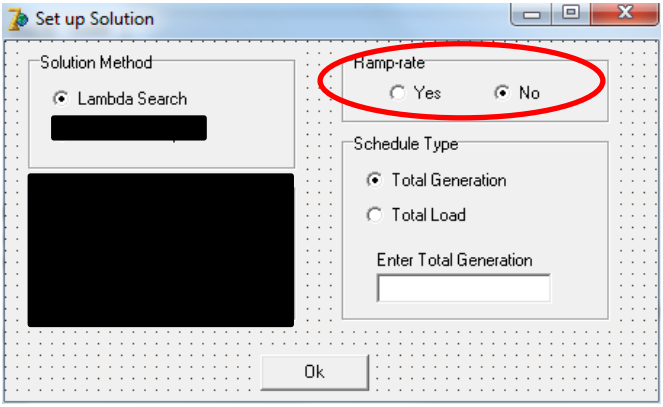
4.1.1 Implementasi Batasan *Ramp-rate*

Untuk mengimplementasikan konsep batasan *ramp-rate* pada perhitungan *economic dispatch*, hal pertama yang dilakukan adalah dengan memberikan tempat untuk mengisikan parameter-parameter *ramp-rate*. Parameter *ramp-rate* yang ditambahkan adalah *down-rate* (DR), *up-rate* (UR), dan pembangkitan sebelumnya (*initial generation*). Pengisian ketiga parameter tersebut saya tambahkan pada tampilan pengisian data pembangkit yang dapat dilihat pada Gambar 4.1 (kondisi sebelum diimplementasikan dapat dilihat pada Gambar 3.3)

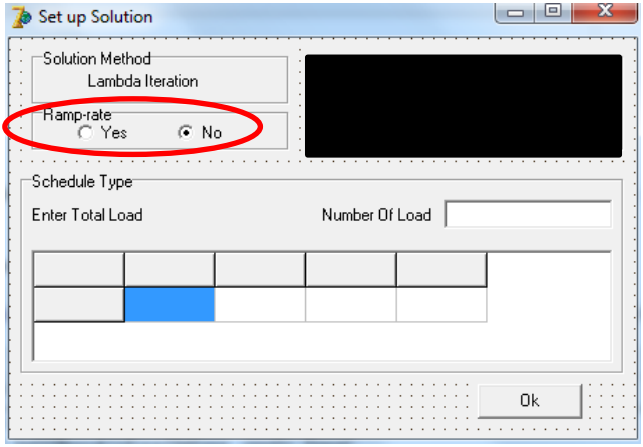


Gambar 4.1 Implementasi *ramp-rate* pada tampilan Data Pembangkit

Selain pada tampilan data pembangkit, saya juga menambahkan pilihan untuk menggunakan batasan *ramp-rate* atau tidak pada tampilan *Set up Solution*. Pilihan ini saya terapkan baik pada menu EDC (*Economic Dispatch*) maupun menu DED (*Dynamic Economic Dispatch*). Tampilan *Set up Solution* yang sudah saya tambahkan dapat dilihat pada Gambar 4.2 dan Gambar 4.3.



Gambar 4.2 Implementasi *ramp-rate* pada tampilan *Set up Solution* menu EDC



Gambar 4.3 Implementasi *ramp-rate* pada tampilan *Set up Solution* menu DED

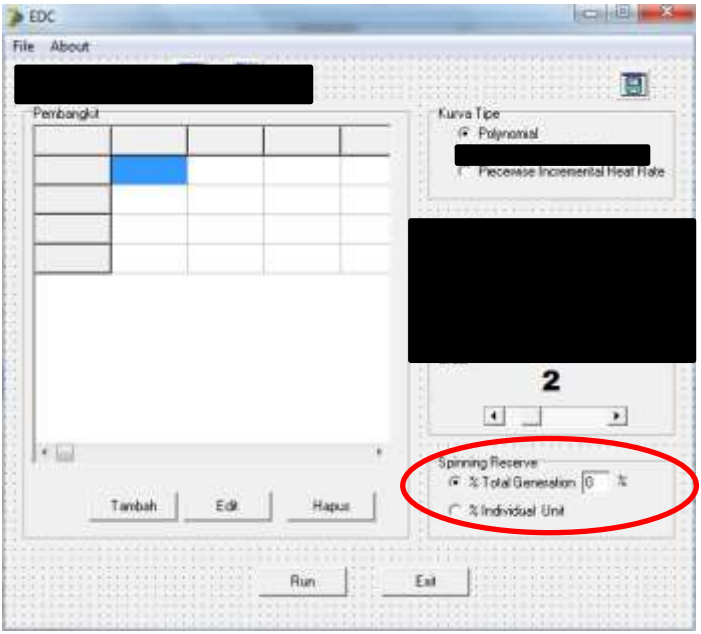
Seperti yang telah dijelaskan sebelumnya, batasan *ramp-rate* akan berpengaruh pada batas pembangkitan maksimum dan minimum setiap unit pembangkit berdasarkan pembangkitan sebelumnya. Berikut merupakan kode logika yang ditambahkan pada program ini.

```
if unitsebelum[i] <> 0 then if ramprate then
  begin
  if (unitsebelum[i]+UR[i])<(unitmax[i]) then pmax[i] := unitsebelum[i]+UR[i];
  if (unitsebelum[i]-DR[i])>(unitmin[i]) then pmin[i] := unitsebelum[i]-DR[i];
  end;
```

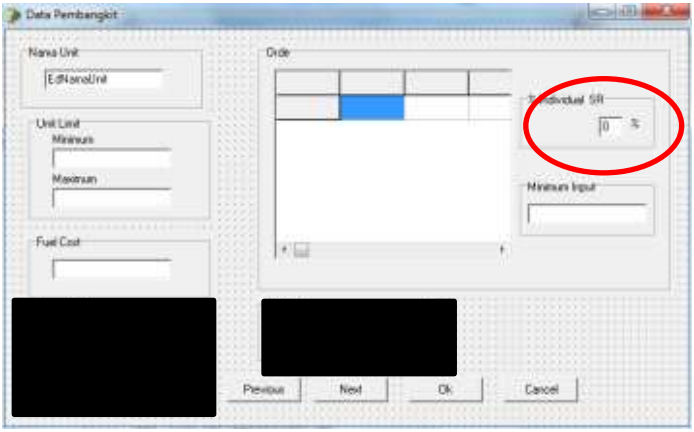
4.1.2 Implementasi Batasan *Spinning Reserve*

Untuk mengimplementasikan konsep batasan *spinning reserve* (SR) pada perhitungan *economic dispatch,* hal pertama yang dilakukan adalah dengan memberikan tempat untuk mengisikan parameter-parameter SR. Pada tugas akhir ini ada dua cara untuk mengisikan batasan SR. Cara pertama adalah dengan mengisikan persentase SR dari total kapasitas pembangkitan seluruh pembangkit. Cara kedua adalah dengan mengisikan persentase SR dari setiap unit pembangkit secara terpisah. User dapat memilih cara pengisian SR pada tampilan utama baik pada menu EDC maupun DED. Setelah diimplementasikan, tampilan menu DED maupun EDC terlihat seperti pada Gambar 4.4

Jika *user* memilih cara pertama, maka *user* dapat langsung mengisikan persentase yang diinginkan pada menu utama. Namun apabila *user* memilih cara kedua, maka *user* memilih % **Individual Unit** dan mengisikan nilai persentasi masing-masing unit pada tampilan Data Pembangkit. Tampilan Data

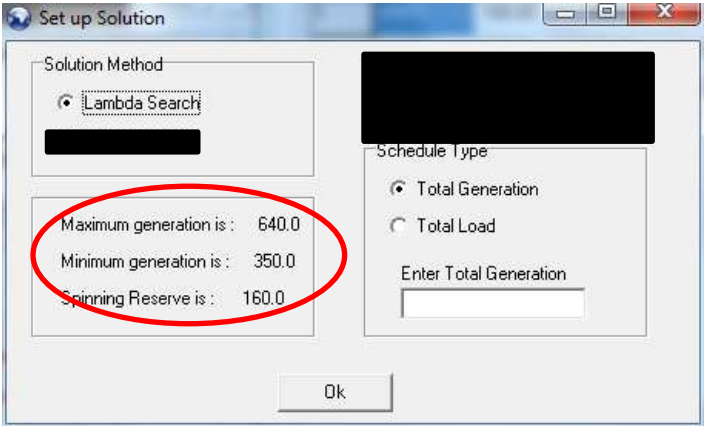


Gambar 4.4 Implementasi *spinning reserve* pada tampilan utama menu DED dan EDC

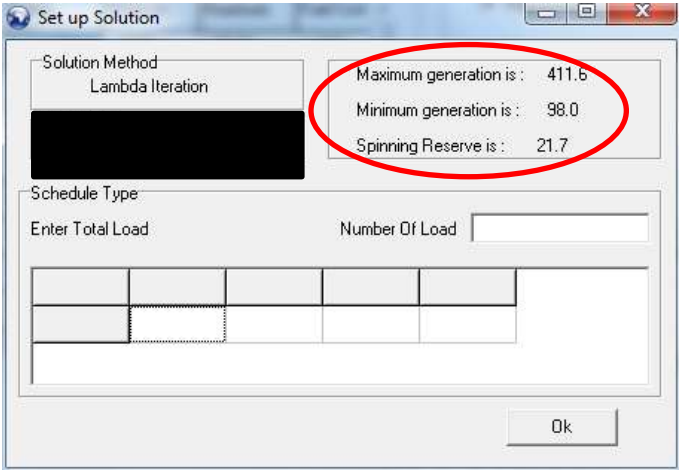


Gambar 4.5 Implementasi *spinning reserve* pada tampilan Data Pembangkit

Selain itu, pada tampilan *Set up Solution user* dapat melihat total maksimum kapasitas pembangkitan setelah adanya batasan SR. Tampilannya dapat dilihat pada Gambar 4.6 dan Gambar 4.7



Gambar 4.6 Implementasi *spinning reserve* pada tampilan *Set up Solution* menu EDC



Gambar 4.7 Implementasi *spinning reserve* pada tampilan *Set up Solution* menu DED

Seperti yang telah dijelaskan sebelumnya, batasan *spinning reserve* akan berpengaruh pada batas pembangkitan maksimum setiap unit pembangkit berdasarkan kapasitas pembangkitannya. Berikut merupakan kode logika yang ditambahkan pada program ini setelah mempertimbangkan *spinning reserve* dan juga *ramp-rate*.

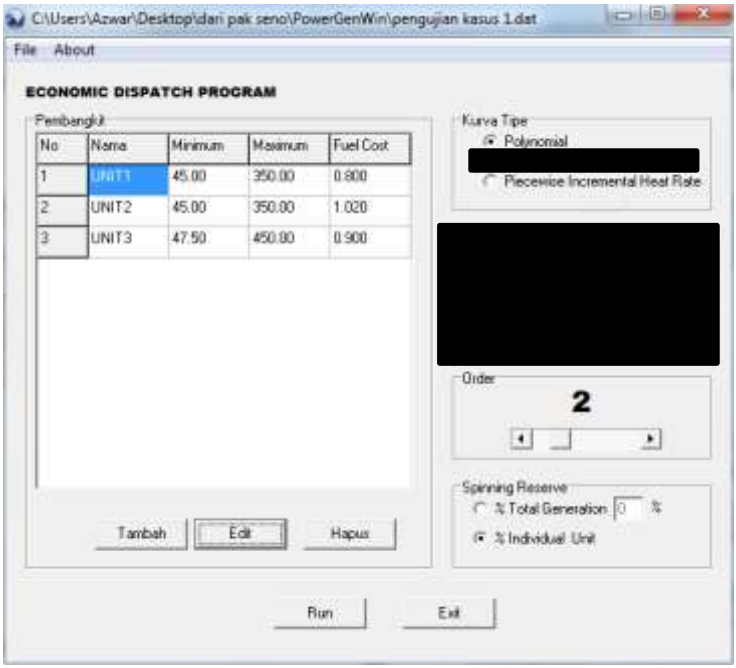
```
for i := 1 to ngen do
begin
pmax[i]:=unitmax[i]-(individuSR[i]/100)*unitmax[i];
pmin[i]:=unitmin[i];
if unitsebelum[i] <> 0 then if ramprate then
  begin
  if (unitsebelum[i]+UR[i])<(pmax[i]) then pmax[i] := unitsebelum[i]+UR[i];
  if (unitsebelum[i]-DR[i])>(pmin[i]) then pmin[i] := unitsebelum[i]-DR[i];
  end;
end;
```

4.2 Hasil Pengujian

Pada sub bab ini akan ditunjukkan hasil perhitungan *software* yang telah dikembangkan. Hasil tersebut kemudian dibandingkan dengan hasil perhitungan manual pada bab 3.3 serta *paper* referensi[1].

4.2.1 Hasil Pengujian Kasus 1

Kasus 1 diuji dengan menggunakan menu EDC. Pada tampilan menu EDC, dipilih tipe kurva **Polynomial** dan **% Individual Unit**, yang terlihat pada Gambar 4.4. Pada tampilan Data Pembangkit, data-data karakteristik unit pembangkit diisikan sesuai data kasus 1, yang telihat pada Gambar 4.5.



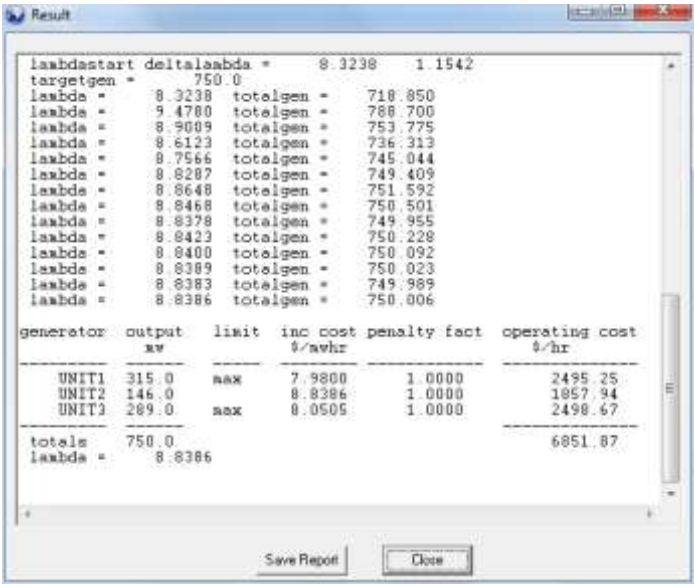
Gambar 4.8 Pengujian kasus 1 tampilan menu EDC

Gambar 4.9 Pengujian kasus 1 tampilan Data Pembangkit

Setelah itu pada tampilan *Set up Solution* – pada Gambar 4.6 – terlihat bahwa pembangkitan maksimum adalah 972.5 MW dengan *spinning reserve* sebesar 177.5 MW. Lalu memilih pilihan **Yes** untuk memperhitungkan *ramp-rate*.

Gambar 4.10 Pengujian kasus 1 tampilan *Set up Solution*

Setelah dijalankan maka tampilan hasil perhitungan akan muncul seperti pada Gambar 4.7.

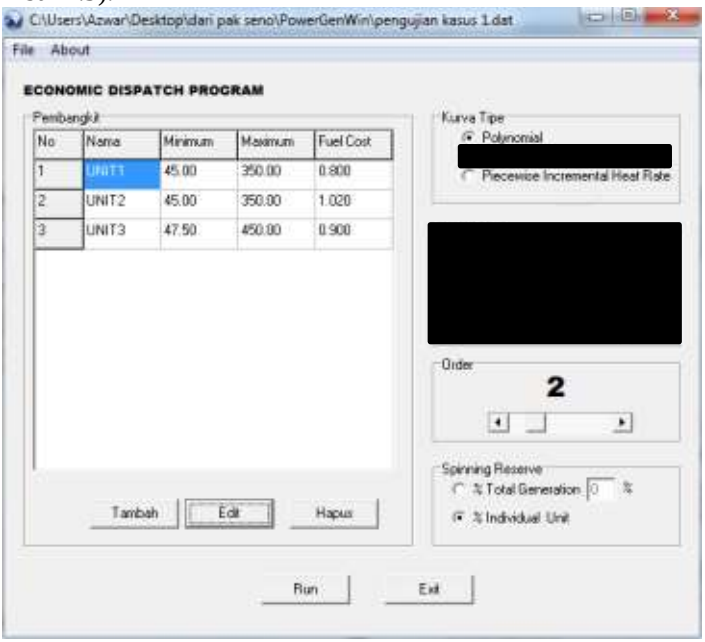


Gambar 4.11 Tampilan hasil perhitungan kasus 1

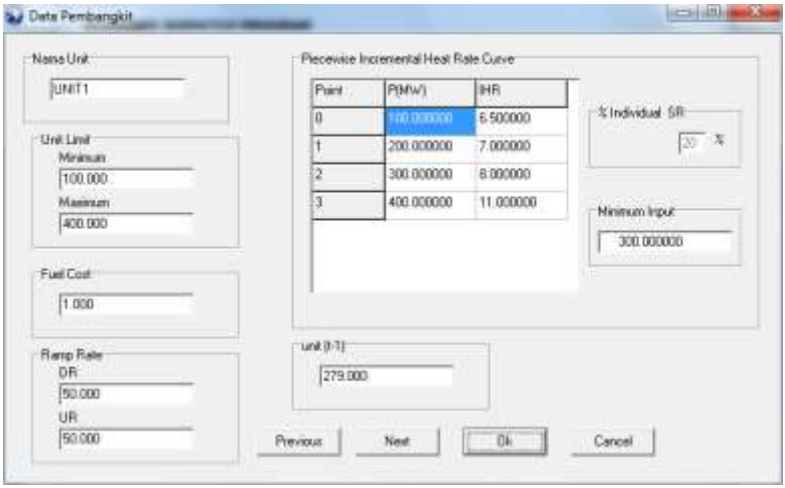
Jika diperhatikan, pembangkitan untuk unit 1 hingga unit 3 sudah sama dengan hasil pada perhitungan manual (Subsub Bab 3.2.1). Oleh karena itu dipastikan hasil perhitungan dari yang dilakukan oleh Powergen sudah benar.

4.2.2 Hasil Pengujian Kasus 2

Kasus 2 juga diuji dengan menggunakan menu EDC. Pada tampilan menu EDC, dipilih tipe kurva **Piecewise Incremental Heat Rate.** Selain itu **% Total Generation** juga dipilih dan diisi "20" pada kolom disampingnya (Gambar 4.8). Pada tampilan Data Pembangkit, data-data karakteristik unit pembangkit diisikan sesuai data kasus 2 (Gambar 4.9).

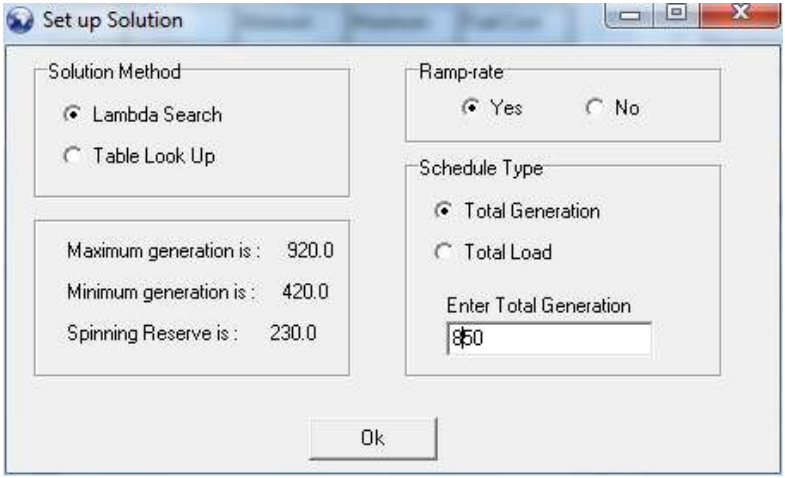


Gambar 4.12 Pengujian kasus 2 tampilan menu EDC



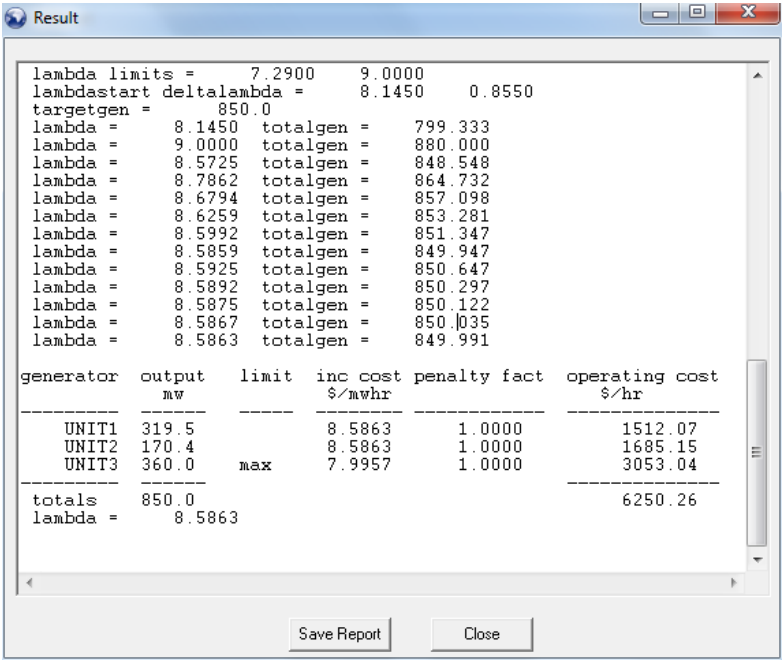
Gambar 4.13 Pengujian kasus 2 tampilan Data Pembangkit

Setelah itu pada tampilan *Set up Solution* terlihat bahwa pembangkitan maksimum adalah 920 MW dengan *spinning reserve* sebesar 230 MW (20% dari total 1150 MW). Lalu memilih pilihan **Yes** untuk memperhitungkan *ramp-rate*.



Gambar 4.14 Pengujian kasus 2 tampilan *Set up Solution*

Setelah dijalankan maka tampilan hasil perhitungan akan muncul seperti pada Gambar 4.11.

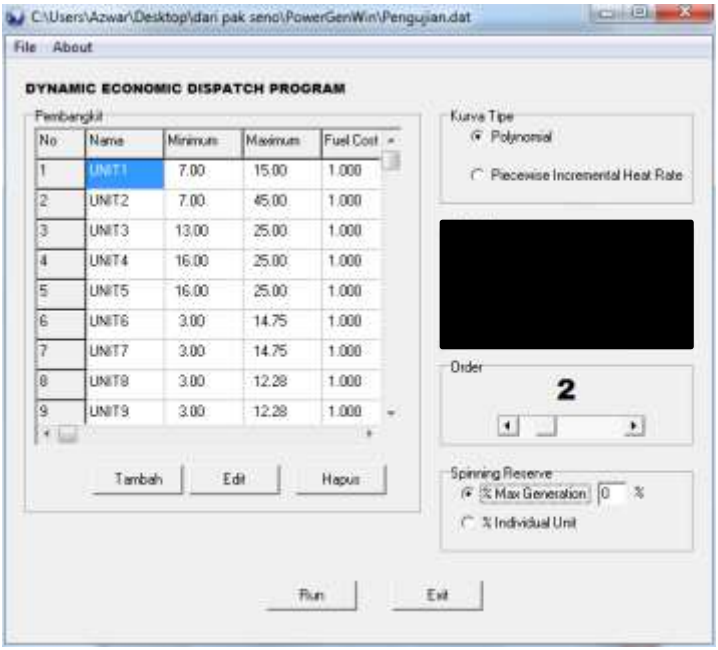


Gambar 4.15 Tampilan hasil perhitungan kasus 2

Jika diperhatikan, pembangkitan untuk unit 1 hingga unit 3 sudah sama dengan hasil pada perhitungan manual (Subsub Bab 3.2.2). Oleh karena itu dipastikan hasil perhitungan dari yang dilakukan oleh Powergen sudah benar.

4.2.3 Hasil Pengujian Kasus 3

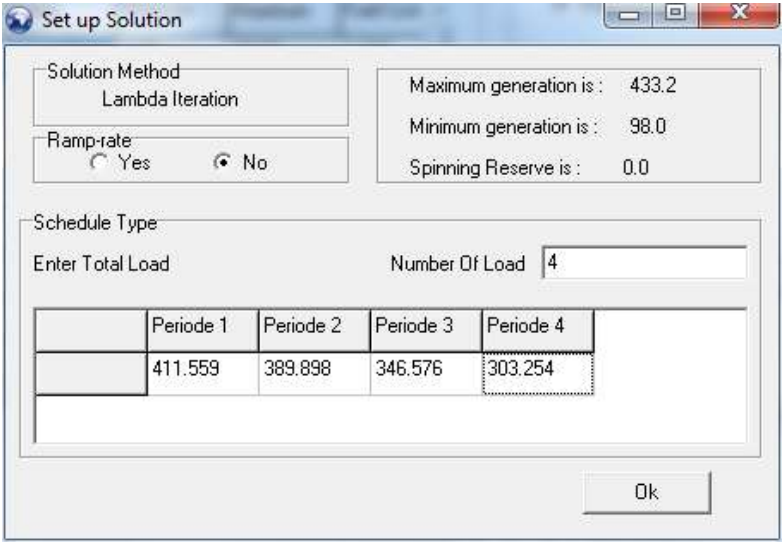
Kasus 3 diuji dengan menggunakan menu DED. Pada tampilan menu DED, dipilih tipe kurva **Polynomial** (Gambar 4.12). Pada tampilan Data Pembangkit, data-data karakteristik unit pembangkit diisikan sesuai data kasus 3 (Gambar 4.13).



Gambar 4.16 Pengujian kasus 3 tampilan menu DED

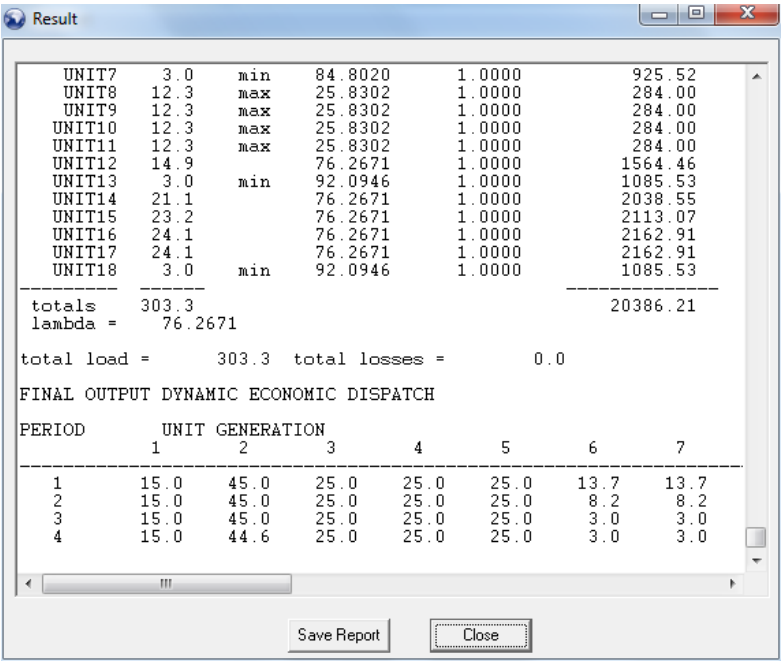
Gambar 4.17 Pengujian kasus 3 tampilan Data Pembangkit

Setelah itu pada tampilan *Set up Solution* dipilih pilihan **No** untuk mengabaikan batasan *ramp-rate*. Lalu pada kolom **Number of Load** diisikan angka 4 untuk menampilan 4 kolom periode pembebanan. Pada kolom tersebut kita bisa mengisikan sejumlah pembebanan sesuai dengan kasus 3.



Gambar 4.18 Pengujian kasus 3 tampilan *Set up Solution*

Setelah dijalankan maka tampilan hasil perhitungan akan muncul seperti pada Gambar 4.15. Hasil akhir dari perhitungan yang dilakukan oleh program Powergen tercantum pada Tabel 4.1. Hasil *output* yang lebih rinci akan dilampirkan pada bagian lampiran.



Gambar 4.19 Tampilan hasil perhitungan kasus 3

Tabel 4.1 Hasil kasus 3

Unit	Time Interval			
	1	**2**	**3**	**4**
1	15.0	15.0	15.0	15.0
2	45.0	45.0	45.0	44.6
3	25.0	25.0	25.0	25.0
4	25.0	25.0	25.0	25.0
5	25.0	25.0	25.0	25.0
6	13.7	8.2	3.0	3.0
7	13.7	8.2	3.0	3.0
8	12.3	12.3	12.3	12.3
9	12.3	12.3	12.3	12.3

Unit	Time Interval			
	1	**2**	**3**	**4**
10	12.3	12.3	12.3	12.3
11	12.3	12.3	12.3	12.3
12	24.0	24.0	20.7	14.9
13	6.4	3.1	3.0	3.0
14	36.2	36.2	30.9	21.1
15	45.0	42.5	32.4	23.2
16	37.0	37.0	33.2	24.1
17	45.0	43.4	33.2	24.1
18	6.4	3.1	3.0	3.0
Biaya	29731.07	27653.74	23855.28	20386.21
Beban	411.6	389.9	346.6	303.3

Berbeda dengan kasus-kasus sebelumnya, kasus 3, 4, dan 5 tidak dibandingkan dengan hasil perhitungan, namun dibandingkan dengan hasil pada referensi[1]. Berikut adalah hasil dari referensi tersebut.

Tabel 4.2 Referensi hasil kasus 3

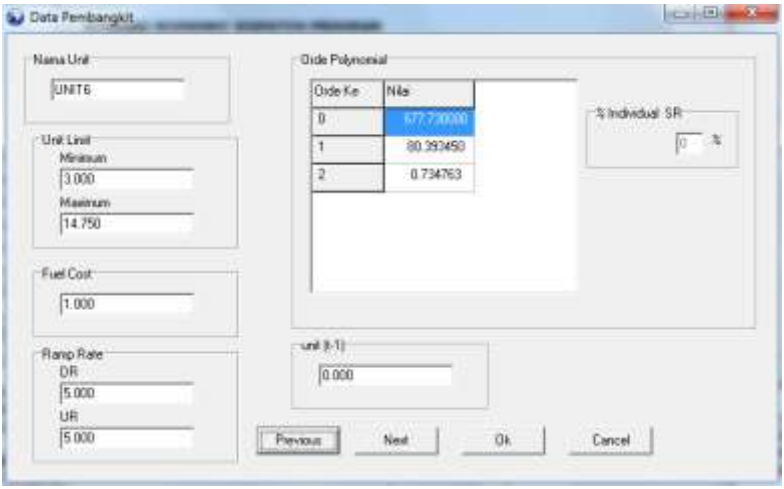
		Time Interval (h)			
		1	2	3	4
Unit	1	15	15	15	15
	2	45	45	45	44.631
	3	25	25	25	25
	4	25	25	25	25
	5	25	25	25	25
	6	13.706	8.213	3	3
	7	13.706	8.213	3	3
	8	12.28	12.28	12.28	12.28
	9	12.28	12.28	12.28	12.28

		Time Interval (h)			
		1	2	3	4
Unit	10	12.28	12.28	12.28	12.28
	11	12.28	12.28	12.28	12.28
	12	24	24	20.726	14.882
	13	6.413	3.149	3	3
	14	36.2	36.2	30.865	21.132
	15	45	42.484227	32.365	23.239
	16	37	37	33.249	24.123
	17	45	43.368839	33.249	24.123
	18	6.4131782	3.1490636	3	3
	Pload	411.559	389.898	346.576	303.254
	Ploss	0	0	0	0
	F($)	29731.066	27653.75	23855.286	20386.215
	λ	100.535	92.463	83.947	76.267

Jika diperhatikan, pembangkitan untuk unit 1 hingga unit 18 sudah sama dengan hasil pada referensi. Oleh karena itu dapat dipastikan hasil perhitungan dari yang dilakukan oleh Powergen sudah benar.

4.2.4 Hasil Pengujian Kasus 4

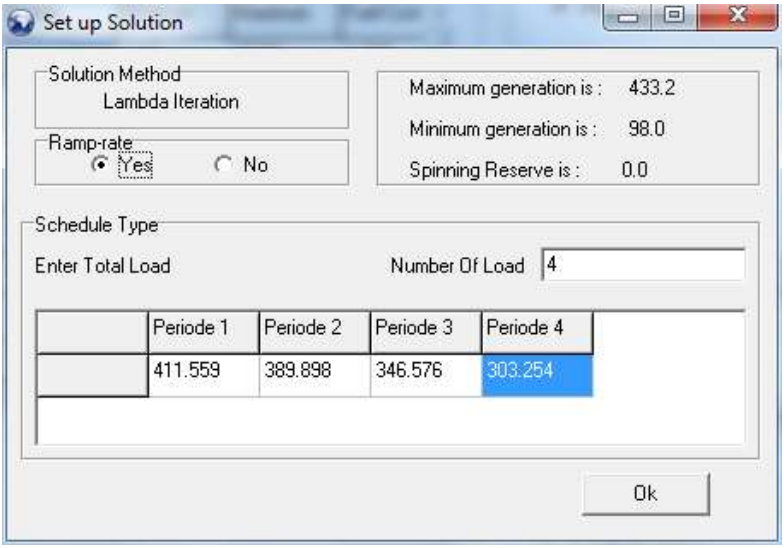
Karena kasus 4 menggunakan data unit pembangkit yang sama, pada kasus ini kita tinggal menambahkan data *up-rate* dan *down-rate* pada tampilan Data Pembangkit (Gambar 4.16).



Gambar 4.20 Pengujian kasus 4 tampilan Data Pembangkit

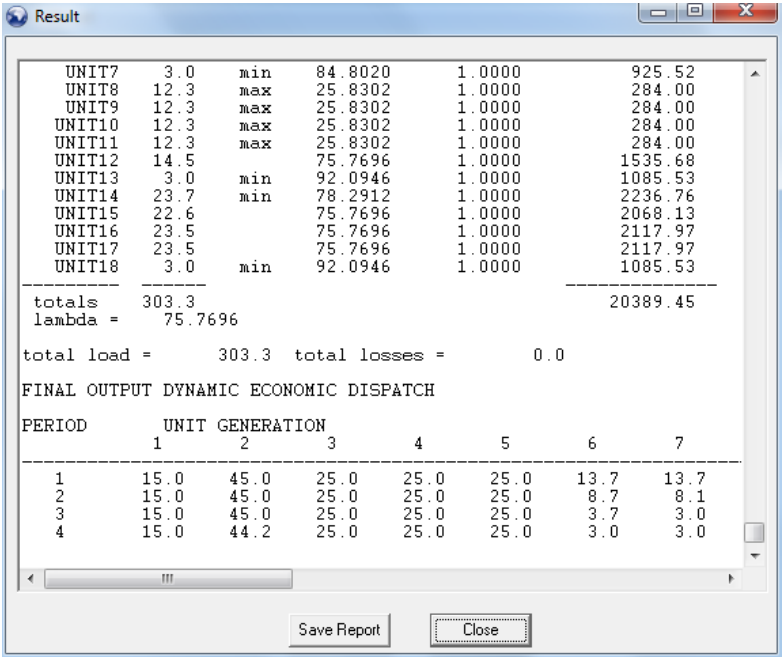
Setelah itu pada tampilan *Set up Solution* dipilih pilihan **Yes** untuk memperhitungkan batasan *ramp-rate* sepeti pada Gambar 4.17

.



Gambar 4.21 Pengujian kasus 4 tampilan *Set up Solution*

Setelah dijalankan maka tampilan hasil perhitungan akan muncul seperti pada Gambar 4.18. Hasil akhir dari perhitungan yang dilakukan oleh program Powergen tercantum pada Tabel 4.3. Hasil *output* yang lebih rinci akan dilampirkan pada bagian lampiran. Terakhir tidak lupa data hasil kasus 4 kembali dibandingkan dengan hasil pada referensi[1].



Gambar 4.22 Tampilan hasil perhitungan kasus 4

Tabel 4.3 Hasil kasus 4

Unit	Time Interval			
	1	**2**	**3**	**4**
1	15.0	15.0	15.0	15.0
2	45.0	45.0	45.0	44.2
3	25.0	25.0	25.0	25.0

Unit	Time Interval			
	1	**2**	**3**	**4**
4	25.0	25.0	25.0	25.0
5	25.0	25.0	25.0	25.0
6	13.7	8.7	3.7	3.0
7	13.7	8.1	3.0	3.0
8	12.3	12.3	12.3	12.3
9	12.3	12.3	12.3	12.3
10	12.3	12.3	12.3	12.3
11	12.3	12.3	12.3	12.3
12	24.0	24.0	20.6	14.5
13	6.4	3.1	3.0	3.0
14	36.2	36.2	30.7	23.7
15	45.0	42.2	32.2	22.6
16	37.0	37.0	33.1	23.5
17	45.0	43.1	33.1	23.5
18	6.4	3.4	3.0	3.0
Biaya	29731.07	27654.1	23856.3	20389.45
Beban	411.6	389.9	346.6	303.3

Tabel 4.4 Referensi hasil kasus 4

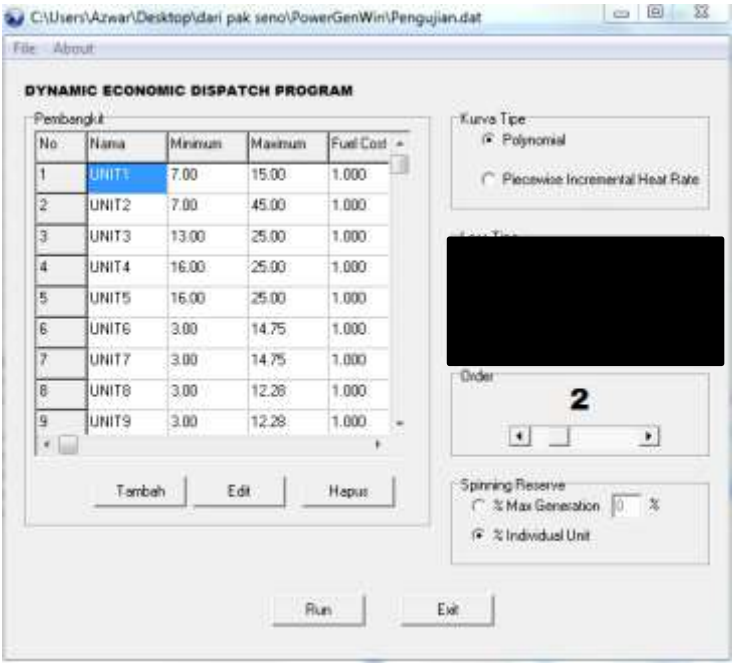
		Time Interval (h)			
		1	2	3	4
Unit	1	15	15	15	15
	2	45	45	45	44.219
	3	25	25	25	25
	4	25	25	25	25
	5	25	25	25	25

		Time Interval (h)			
		1	2	3	4
Unit	6	13.706	8.706	3.706	3
	7	13.706	8.064	3	3
	8	12.28	12.28	12.28	12.28
	9	12.28	12.28	12.28	12.28
	10	12.28	12.28	12.28	12.28
	11	12.28	12.28	12.28	12.28
	12	24	24	20.625	14.503
	13	6.413	3.06	3	3
	14	36.2	36,2	30.697	23.697
	15	45	42.224	32.224	22.648
	16	37	37	33.092	23.532
	17	45	43.108	33.108	23.532
	18	6.413	3.413	3	3
	Pload	411.559	389.898	346.576	303.254
	Ploss	0	0	0	0
	F($)	29731.066	27654.098	23856.301	20389.449
	λ	100.535	92.244	83.828	75.769

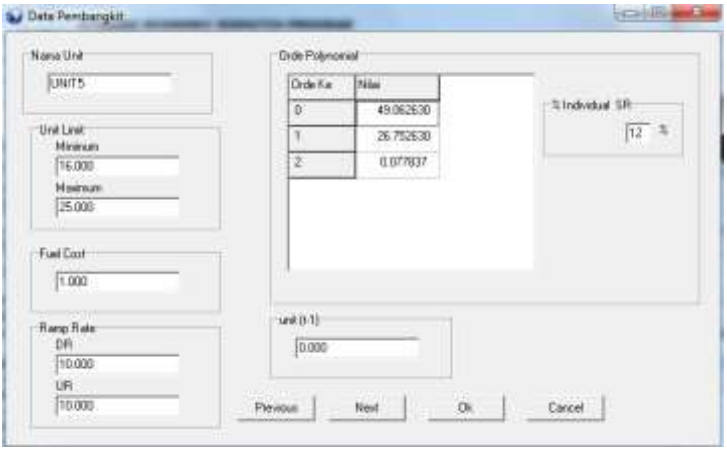
Jika diperhatikan, pembangkitan untuk unit 1 hingga unit 18 sudah sama dengan hasil pada referensi. Oleh karena itu dapat dipastikan hasil perhitungan dari yang dilakukan oleh Powergen sudah benar.

4.2.5 Hasil Pengujian Kasus 5

Kasus 5 menggunakan data unit pembangkit yang sama, pada kasus ini kita tinggal memilih **% Individual Unit** pada tampilan menu DED (Gambar 4.19) serta menambahkan sejumlah persentase minimum *spinning reserve* sesuai data kasus 5 pada tampilan Data Pembangkit (Gambar 4.20).

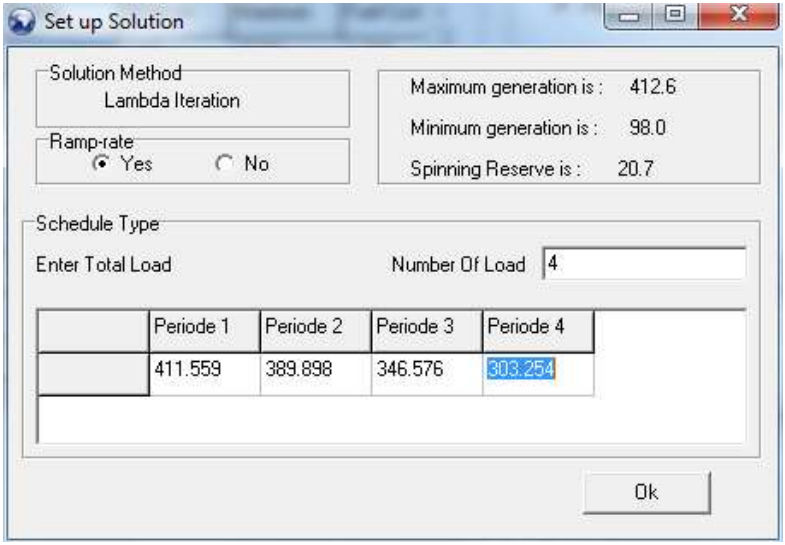


Gambar 4.23 Pengujian kasus 5 tampilan menu DED



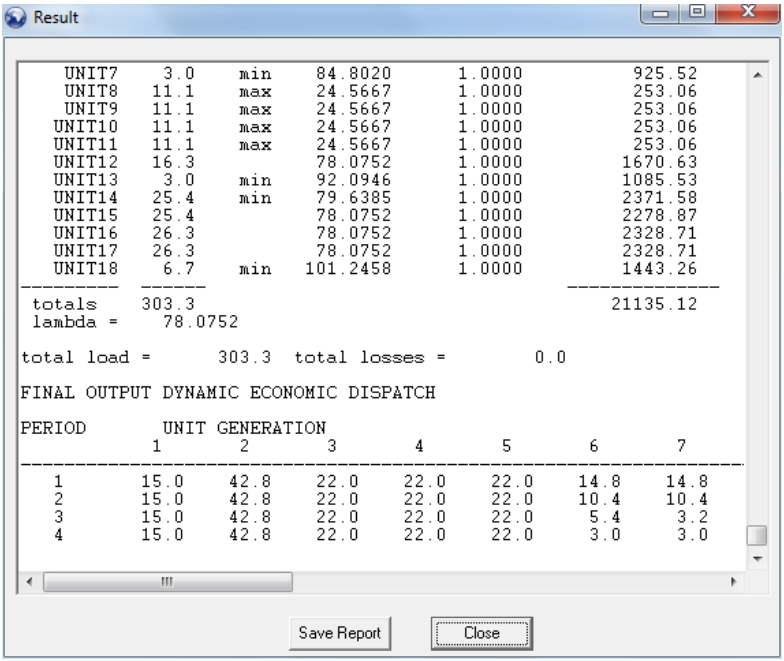
Gambar 4.24 Pengujian kasus 3 tampilan Data Pembangkit

Tidak ada yang perlu diubah pada tampilain *Set up Solution*, hanya saja kali ini bisa kita lihat pembangkitan maksimum yang bisa dilakukan sudah berkurang menjadi 412.6 MW (Gambar 4.21). Berkurangnya kapasitas maksimum ini terjadi akibat batasan *spinning reserve* sebesar 20.7 MW yang harus dipertimbangkan.



Gambar 4.25 Pengujian kasus 5 tampilan *Set up Solution*

Selanjutnya setelah dijalankan maka tampilan hasil perhitungan akan muncul seperti pada Gambar 4.22. Hasil akhir dari perhitungan yang dilakukan oleh program Powergen tercantum pada Tabel 4.5. Hasil *output* yang lebih rinci akan dilampirkan pada bagian lampiran.



Gambar 4.26 Tampilan hasil perhitungan kasus 5

Tabel 4.5 Hasil kasus 5

Unit	Time Interval			
	1	**2**	**3**	**4**
1	15.0	15.0	15.0	15.0
2	42.8	42.8	42.8	42.8
3	22.0	22.0	22.0	22.0
4	22.0	22.0	22.0	22.0
5	22.0	22.0	22.0	22.0
6	14.8	10.4	5.4	3.0
7	14.8	10.4	3.2	3.0
8	11.1	11.1	11.1	11.1
9	11.1	11.1	11.1	11.1

Unit	Time Interval			
	1	**2**	**3**	**4**
10	11.1	11.1	11.1	11.1
11	11.1	11.1	11.1	11.1
12	24.0	24.0	21.7	16.3
13	15.7	5.7	3.0	3.0
14	36.2	36.2	32.4	25.4
15	42.8	42.8	33.8	25.4
16	37.0	37.0	34.7	26.3
17	42.8	42.8	34.7	26.3
18	15.7	12.7	9.7	6.7
Biaya	31029.97	28702.75	24759.32	21135.12
Beban	411.6	389.9	346.6	303.3

Berbeda dengan kasus 3 dan kasus 4, kasus 5 tidak dibandingkan hasil pada referensi. Karena dari kasus-kasus sebelumnya ketepatan perhitungan Powergen sudah benar, maka untuk kasus kali ini pun dianggap sudah benar.

Jika diperhatikan, pembangkitan untuk unit 1 hingga unit18 terdapat banyak perbedaan yang diakibatkan oleh *spinning reserve*. Perbedaan terjadi terutama pada unit-unit yang pada kasus sebelumnya di set pada pembangkitan yang mendekati maksimum.

BAB 5
PENUTUP

2.1 Kesimpulan

Dari hasil pengembangan *software* yang telah dilakukan dapat disimpulkan beberapa hal sebagai berikut :

1. *Software* yang dihasilkan menggunakan bahasa pemrograman Delphi telah mampu untuk melakukan perhitungan *economic dispatch* yang mempertimbangkan batasan *ramp-rate* dan *spinning reserve*, serta karakteristik tidak linier dan fungsi linier *piecewise* dengan benar. Hal ini dibuktikan dengan mencocokkan hasil *software* dengan perhitungan manual.
2. Khusus untuk batasan *ramp-rate* diperlukan simulasi dengan interval pembebanan lebih dari satu karena dibutuhkan *schedule* pembangkitan periode sebelumnya.
3. Metode optimasi iterasi lambda yang digunakan telah mampu melakukan perhitungan *economic dispatch* untuk tipe kurva polinomial dengan orde lebih dari dua maupun *piecewise* dengan lebih dari dua fungsi linier parsial.
4. Dari hasil pengujian *software* diketahui bahwa dengan mempertimbangkan batasan *ramp-rate* bisa saja meningkatkan biaya produksi untuk memenuhi batasan tersebut. Pada sistem 18 unit terlihat jika tanpa mempertimbangkan batasan apapun maka akan didapatkan total biaya pembangkitan sebesar $101,626.3 , bila mempertimbangkan *ramp-rate* didapatkan biaya pembangkitan sebesar $101,630.92.
5. Perhitungan dengan mempertimbangkan batasan *spinning reserve* pun bisa saja meningkatkan biaya produksi. Dari hasil pengujian jika menambahkan batasan *spinning reserve* maka biaya pembangkitan naik menjadi $105,627.16
6. Hasil akhir dari *software* yang dikembangkan dapat digunakan untuk meng-*upgrade software* yang selama ini digunakan pada mata kuliah Operasi Optimum Sistem Tenaga Listrik.

2.2 Saran

Saran yang diberikan setelah selesainya tugas akhir ini adalah sebagai berikut :

1. Diperlukan pengujian lebih lanjut dengan sistem yang lebih kompleks (contohnya sistem Jawa-Bali).
2. Dapat ditambahkan metode perhitungan optimasi lainnya – seperti *quadratic programming* – pada *software* powergen.

DAFTAR PUSTAKA

[1] F. Benhamida, et all "Solving Dynamic Economic Load Dispatch With Ramp Rate Limit Using Quadratic Programming" IEEE 2013

[2] Ching-Tzong Su, Chi-Min Lin, and Yung-Fu Wang "Economic Dispatch and Spinning Reserve Scheduling for Generation Transmission Systems" IEEE MELECON 2004, May 12-15,2004, Dubrovnik, Croatia

[3] Ringlee, R.J. and Williams, D.D. "Economic System Operation Considering Valve Throttling Losses II-Distribution of System Loads by the Method of Dynamic Programming" Power Apparatus and Systems, Part III. Transactions of the American Institute of Electrical Engineers, 1962

[4] Hadi S., "Power System Analysis 2nd Edition",McGrowHill, Ch1,1999.

[5] Penangsang, O., "Analisis Aliran Daya", ITS Press Surabaya, 2012

[6] Wibowo, R. S., Nursidi, Satriyadi H, I. G. N., Uman P, D. F., Soeprijanto, A., Penangsang, O., "Dynamic DC Optimal Power Flow using Quadratic Programming", International Conference on Information Technology and Electrical Engineering (ICITEE), 360-364,2013

[7] Wood, A. J. dan Wollenberg, B. F., "Power Generation, Operation and Control", Wiley., New York, 2nd ed., 1996

[8] Aristyo, M. Fauzan, "Online Simulator Untuk Operasi Optimum Sistem Tenaga Listrik (Dynamic Unit Commitment Economic Dispatch - Optimal Power Flow)" Tugas Akhir Jurusan Teknik Elektro Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember, 2014

[9] Wardana, Rizkiananto, "Implementasi Algoritma Particle Swarm Optimization Untuk Permasalahan Dynamic Economic Dispatch Dengan Batasan Ramp-Rate dan Valve Point Efect", Tugas Akhir Jurusan Teknik Elektro Fakultas Teknologi Industri Institut Teknologi Sepuluh Nopember, 2014

[10] Trisiana, Yunitika, "Optimisasi Economic Dispatch Menggunakan Imperialist Competitive Algorithm (Ica) Pada Sistem Tenaga Listrik", Tugas Akhir Jurusan Teknik Elektro Fakultas Teknologi Industri Institut Teknologi Sepuluh Nopember, 2011

[11] Kusnassriyanto, “Belajar Pemrograman Delphi”, Penerbit Modula Bandung, Bandung, 2011
[12] Borland Delphi 7, “Developer’s Guide”, Borland Software Corporation, 2002

LAMPIRAN

Kode logika (*Logic code)*

Unit 1

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Menus,unit3, Grids;

type
  TFrmEdcMain = class(TForm)
    MainMenu1: TMainMenu;
    MnFile: TMenuItem;
    MnNew: TMenuItem;
    MnLoad: TMenuItem;
    N1: TMenuItem;
    MnAbout: TMenuItem;
    GroupBox1: TGroupBox;
    GroupBox2: TGroupBox;
    RbPoly: TRadioButton;
    RbPIO: TRadioButton;
    RbPINC: TRadioButton;
    GroupBox3: TGroupBox;
    RbNoLoss: TRadioButton;
    RbCONSTPF: TRadioButton;
    RbLossFOrm: TRadioButton;
    Order: TGroupBox;
    ScrOrder: TScrollBar;
    LbOrder: TLabel;
    BtTambah: TButton;
    BTEdit: TButton;
    Button2: TButton;
    Button1: TButton;
    Label1: TLabel;
    ODLoadData: TOpenDialog;
    SgNamaPembangkit: TStringGrid;
    BtHapus: TButton;
    Label2: TLabel;
    SDSaveData: TSaveDialog;
    BtLoss: TButton;
    GroupBox4: TGroupBox;
    RbSRPercentage: TRadioButton;
    RbIndividu: TRadioButton;
    edtSR: TEdit;
    Label3: TLabel;

    procedure Button1Click(Sender: TObject);
    procedure ScrOrderChange(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure RbPIOClick(Sender: TObject);
    procedure RbPolyClick(Sender: TObject);
    procedure RbPINCClick(Sender: TObject);
    procedure RbNoLossClick(Sender: TObject);
    procedure RbCONSTPFClick(Sender: TObject);
    procedure RbLossFOrmClick(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure BtTambahClick(Sender: TObject);
    procedure MnLoadClick(Sender: TObject);
    procedure LsNameDblClick(Sender: TObject);
    procedure SgNamaPembangkitDblClick(Sender: TObject);
    procedure BTEditClick(Sender: TObject);
    procedure BtHapusClick(Sender: TObject);
    procedure N1Click(Sender: TObject);
    procedure MnNewClick(Sender: TObject);
    procedure BtLossClick(Sender: TObject);
    procedure edtSRChange(Sender: TObject);
    procedure RbSRPercentageClick(Sender: TObject);
    procedure RbIndividuClick(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FrmEdcMain: TFrmEdcMain;
  tmsg:string;
implementation
```

```
uses Unit2, Unit4, Unit6, Unit7, Unit5;

{$R *.dfm}

function CekGenName(namaGenerator:string):integer;
var s1,s2:string;
  i:integer;
begin
cekGenName:= 0;
s1:=trim(uppercase(namagenerator));
for i:=1 to ngen do
 begin
   s2:=trim(uppercase(genname[i]));
   if s2 = s1 then cekGenname := i;
 end;
end;
Procedure GetNewGeneratorName(var GName:string);
var i: integer;
    s1,s2: string;
begin
   i := 0;
   repeat
     i:=i+1;
     str(i,s1);
     s2 :='UNIT'+TRim(s1);
   until cekGenname(s2) = 0;
   gname:=s2;
end;

procedure TFrmEdcMain.Button1Click(Sender: TObject);
label keluar;
var d,e,f,s:string;
    i,ng,Od:integer;
begin
if RbSRPercentage.Checked then
begin
for i:=1 to ngen do IndividuSR[i]:=SR;
end;
if ngen = 0 then   showmessage('Tidak ada pembangkit');
  if ngen>0 then
  begin
    if curvetype =  pio then
    begin
       if not(cekiopoint(ng,od)) then
       begin
          str(od,s);
         showmessage('Error on '+genname[ng]+'  Point '+s);
         goto keluar;
       end;
    end;
    if curvetype =  pinc then
    begin
       if not(cekihrpoint(ng,od)) then
       begin
          str(od,s);
         showmessage('Error on '+genname[ng]+'  Point '+s);
         goto keluar;
       end;
    end;


    diagflag := true;
    FrmEdcSetupSolution.showmodal;
    if prosesrun then FrmEdcResult.showmodal;
  end;
  keluar:
end;

procedure TFrmEdcMain.ScrOrderChange(Sender: TObject);
var s:string;
    k:integer;
begin
  str(scrorder.Position ,s);
  lborder.Caption := s ;
   curveorder:=scrorder.Position ;
end;



procedure TFrmEdcMain.FormCreate(Sender: TObject);
begin
ngen := 0 ;
losstype :=noloss;
curvetype :=poly;
curveorder:=2;
SgNamaPembangkit.RowCount := 2;
SgNamaPembangkit.Colwidths[0]:=40;
SgNamaPembangkit.cells[0,0]:='No';
```

```
SgNamaPembangkit.cells[1,0]:='Nama';
SgNamaPembangkit.cells[2,0]:='Minimum';
SgNamaPembangkit.cells[3,0]:='Maximum';
SgNamaPembangkit.cells[4,0]:='Fuel Cost';
btloss.Enabled :=false;
    filename :='NewEdcFile.dat';


end;


procedure TFrmEdcMain.RbPIOClick(Sender: TObject);
begin
curvetype :=  pio;
end;

procedure TFrmEdcMain.RbPolyClick(Sender: TObject);
begin
curvetype :=poly;
end;

procedure TFrmEdcMain.RbPINCClick(Sender: TObject);
begin
curvetype :=  PINC;
end;

procedure TFrmEdcMain.RbNoLossClick(Sender: TObject);
var i:integer;
begin
penfacd:=penfac;
for i:=1 to ngen do penfac[i]:=1;

btloss.Enabled :=false;
losstype :=noloss;

end;

procedure TFrmEdcMain.RbCONSTPFClick(Sender: TObject);
begin
penfac :=penfacd;
btloss.Enabled :=true;
btloss.Caption := 'Edit Penalty Factor';
losstype :=constpf;
end;

procedure TFrmEdcMain.RbLossFOrmClick(Sender: TObject);
begin
penfac:=penfacd;
btloss.Enabled :=true;
 btloss.Caption := 'Edit Loss Matrix';
losstype:=lossform;
end;

procedure TFrmEdcMain.Button2Click(Sender: TObject);
begin
close;
end;

procedure TFrmEdcMain.BtTambahClick(Sender: TObject);
var i:integer;
begin
  modedatapembangkit:=1;
  NoGenerator:=ngen+1;
  if NoGenerator>20 then
  begin
    showmessage('Jumlah Maximum Pembangkit 20');
  end else
  begin
    getNewGeneratorName(genname[NoGenerator]);
    penfac[noGenerator]:=1.0;
    fuelcost[nogenerator]:=1.0;
    frmedcdataPembangkit.showmodal;
    pgenmax:=0;
    pgenmin:=0;
    for i := 1 to ngen do
       begin
        ihr_ftn( i, pmax[ i ], maxihr[ i ] );
        ihr_ftn( i, pmin[ i ], minihr[ i ] );
        pgenmax := pgenmax + pmax[ i ];
        pgenmin := pgenmin + pmin[ i ];
      end;
    end;
end;

procedure TFrmEdcMain.MnLoadClick(Sender: TObject);
 label keluar;
 var i,k:integer;
 s:string;
begin
if OdLoaddata.Execute then
```

```
begin

 filename:= ODLoadData.filename;
 if not(cekedcfile(filename )) then
 begin
   showmessage('Bukan File EDC ');
   goto keluar;
 end;

 frmedcmain.caption := filename;

 datainput(filename);
 penfacd:=penfac;

 SgNamaPembangkit.RowCount := ngen+1;
 SgNamaPembangkit.ColWidths[0]:=40;

 for i:=1 to ngen do
 begin
   unitmax[i]:=pmax[i];
   unitmin[i]:=pmin[i];
   str(i,s);
   sgnamapembangkit.cells[0,i]:= s;
   sgnamapembangkit.cells[1,i]:= genname[i] ;

   str(pmin[i]:7:2,s);
   sgnamapembangkit.cells[2,i]:= s;
   str(pmax[i]:7:2,s);
   sgnamapembangkit.cells[3,i]:= s;
   str(fuelcost[i]:6:3,s);
   sgnamapembangkit.cells[4,i]:= s;

 end;
 if curvetype =poly then
 begin
  FrmEdcMain.rbpoly.Checked :=true;
 end;
 if curvetype =pinc then
 begin
   FrmEdcMain.rbpinc.Checked :=true;
 end;
 if curvetype =pio then
 begin
   FrmEdcMain.rbpio.Checked :=true;
 end;
 if losstype =noloss then frmEdcMain.RbNoLoss.Checked :=true;
 if losstype =lossform then frmEdcMain.RbLossForm.Checked := true;
 if losstype =constpf then frmEdcMain.RbCONSTPF.Checked :=true;
 frmEdcMain.scrorder.Position := curveorder;
 frmEdcMain.Caption :=filename;
  keluar:
 end;


end;

procedure TFrmEdcMain.LsNameDblClick(Sender: TObject);
var i:integer;
begin
{  if lsname.Count >0 then
  begin
        NoGenerator:=lsname.ItemIndex+1 ;
        FrmEdcDataPembangkit.showmodal;


  end;}
end;
procedure DUMM;
Begin
{  if lsname.Count >0 then
   begin
        NoGenerator:=lsname.ItemIndex+1 ;
        FrmEdcDataPembangkit.showmodal;


  end;
end; }

end;

procedure TFrmEdcMain.SgNamaPembangkitDblClick(Sender: TObject);
begin
modedataPembangkit:=0;
if (sgnamapembangkit.Row >0) and (sgnamapembangkit.row<=ngen )then
  begin
        NoGenerator:=sgnamapembangkit.Row ;
        FrmEdcDataPembangkit.showmodal;
 end;

end;
```

```
procedure TFrmEdcMain.BTEditClick(Sender: TObject);
begin
if ngen = 0 then   showmessage('Tidak ada pembangkit');
if ngen>0 then
begin
modedataPembangkit:=0;
if (sgnamapembangkit.Row >0) and (sgnamapembangkit.row<=ngen )then
  begin
       NoGenerator:=sgnamapembangkit.Row ;
       FrmEdcDataPembangkit.showmodal;
 end;
 end ;
end;

procedure TFrmEdcMain.BtHapusClick(Sender: TObject);
var n:integer;
   s:string;
   i,j:integer;
begin
if ngen = 0 then   showmessage('Tidak ada pembangkit');

  if ngen>=1 then
  begin
    noGenerator:=sgNamaPembangkit.Row ;
    if (NoGenerator>0) and (NoGenerator<=nGen) Then
    begin
       s:= genname[NoGenerator]+' Dihapus ?';;
       n:= MessageDlg(s, mtConfirmation         ,[mbOK,mbCancel], 0 );
       if n= mrOk then
       begin
         if Ngen  = 1 then
         begin
            Ngen:= 0;
         end else
         begin
           for i:= NoGenerator to ngen-1 do
           begin
             genname[i]:=genname[i+1];
             penfac[i]:=penfac[i+1];
             pmax[i]:=pmax[i+1];
             pmin[i]:=pmin[i+1];
             fuelcost[i]:=fuelcost[i+1];
             for j:=0 to curveorder do
             begin
               coeff[i,j] :=coeff[i+1,j];
               ihr_mwpoint[i,j]:=ihr_mwpoint[i+1,j];
               ihr_cost[i,j] :=ihr_cost[i+1,j];
               io_mwpoint[i,j]:=io_mwpoint[i+1,j];
               io_cost[i,j]:=io_cost[i+1,j];
             end;
             b0[ i ] :=b0[i+1];
           end;
           for i := NoGenerator to ngen-1 do
           begin
             for j := 1 to ngen  do
             begin
                 b[ i, J ]:= b[ i+1, J ];
             end;
           end;
           for i := 1 to ngen-1 do
           begin
             for j := NoGenerator to ngen-1  do
             begin
                 b[ i, J ]:= b[ i, J+1 ];
             end;
           end;

           ngen := ngen-1;
           pgenmax:=0;
           pgenmin:=0;
           for i := 1 to ngen do
           begin
              ihr_ftn( i, pmax[ i ], maxihr[ i ] );
              ihr_ftn( i, pmin[ i ], minihr[ i ] );
              pgenmax := pgenmax + pmax[ i ];
              pgenmin := pgenmin + pmin[ i ];
           end;
         end;
         if ngen >1 then
         begin
            SgNamaPembangkit.RowCount := ngen+1;
         end else
         begin
            SgNamaPembangkit.RowCount := 2;
            sgnamapembangkit.cells[0,1]:= '';
            sgnamapembangkit.cells[1,1]:= '';
            sgnamapembangkit.cells[2,1]:= '';
            sgnamapembangkit.cells[3,1]:= '';
            sgnamapembangkit.cells[4,1]:= '';
```

```
         end;
         for i:=1 to ngen do
         begin
              str(i,s);
              sgnamapembangkit.cells[0,i]:= s;
              sgnamapembangkit.cells[1,i]:= genname[i] ;

              str(pmin[i]:7:2,s);
              sgnamapembangkit.cells[2,i]:= s;
              str(pmax[i]:7:2,s);
              sgnamapembangkit.cells[3,i]:= s;
              str(fuelcost[i]:6:3,s);
              sgnamapembangkit.cells[4,i]:= s;
         end;
       end;
    end;
  end;
end;
procedure TFrmEdcMain.N1Click(Sender: TObject);
var d,e,f,s:string;
begin

  getfilename(filename,d,f,e);
  sdsavedata.FileName := f+'_Copy' ;
if sdsavedata.Execute then
begin
  s:=trim(sdsavedata.filename);
  getfilename(s,d,f,e);
  if trim (e) = '' then s:=s+'.dat';
  frmEdcMain.Caption :=s;
  dataoutput(s );
end;
end;

procedure TFrmEdcMain.MnNewClick(Sender: TObject);
begin
  ngen:=0;
  SgNamaPembangkit.RowCount := 2;
  sgnamapembangkit.cells[0,1]:= '';
  sgnamapembangkit.cells[1,1]:= '';
  sgnamapembangkit.cells[2,1]:= '';
  sgnamapembangkit.cells[3,1]:= '';
  sgnamapembangkit.cells[4,1]:= '';
  filename:='Noname.dat'
end;

procedure TFrmEdcMain.BtLossClick(Sender: TObject);
begin
  if ngen>0 then
  begin
    if losstype =constpf then  frmedcpenfac.showmodal;

    if  losstype=lossform then frmedclossform.ShowModal;
  end;
end;

procedure TFrmEdcMain.edtSRChange(Sender: TObject);
var k:integer;
    s:string;
begin
s:=trim(FrmEdcMain.edtSR.Text);
val(s,SR,k);
end;

procedure TFrmEdcMain.RbSRPercentageClick(Sender: TObject);
var i:integer;
begin
edtSR.Enabled := true;
FrmedcDataPembangkit.edIndividuSR.Enabled:=false;
end;

procedure TFrmEdcMain.RbIndividuClick(Sender: TObject);
begin
edtSR.Enabled := false;
FrmedcDataPembangkit.edIndividuSR.Enabled:=true;
end;

end.
```

Unit 2

```
unit Unit2;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Grids,unit3;

type
```

```
  TfrmEdcLossForm = class(TForm)
    SgB: TStringGrid;
    SgB0: TStringGrid;
    EdB00: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    BtOk: TButton;
    BtCancel: TButton;
    procedure FormActivate(Sender: TObject);
    procedure BtCancelClick(Sender: TObject);
    procedure BtOkClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmEdcLossForm: TfrmEdcLossForm;

implementation

{$R *.dfm}

procedure TfrmEdcLossForm.FormActivate(Sender: TObject);
var i,j:integer;
    s:string;
begin
  sgb0.ColCount :=ngen+1;
  sgb.ColCount :=ngen+1;
  sgb.rowCount :=ngen+1;
  sgb0.FixedCols :=1;
  sgb0.Fixedrows :=1;
  sgb.FixedCols :=1;
  sgb.Fixedrows :=1;
  str(b00:15:5,s);

   edb00.Text:=trim(s);
  for i:=1 to ngen do
  begin
    str(i,s);
    sgb0.Cells[i,0]:='B'+trim(s);
    sgb.Cells[i,0]:='B'+trim(s);
    sgb.Cells[0,i]:='B'+trim(s);
  end;
  for i:=1 to ngen do
  begin
    str(b0[i]:15:5,s);
    sgb0.Cells[i,1]:=trim(s);
    for j:=1 to ngen do
    begin
       str(b[i,j]:15:5,s);
       sgb.Cells[j,i]:=trim(s);
    end;
  end;


end;

procedure TfrmEdcLossForm.BtCancelClick(Sender: TObject);

begin



close;
end;

procedure TfrmEdcLossForm.BtOkClick(Sender: TObject);
var i,j,k:integer;

begin
  val(trim( edb00.Text),b00,k);
  for i:=1 to ngen do
  begin
    val(trim(sgb0.Cells[i,1]),b0[i],k);
    for j:=1 to ngen do
    begin
      val(trim(sgb.Cells[j,i]),b[i,j],k);

    end;
  end;

close;
end;

end.
```

Unit 3

```
unit Unit3;

interface
uses  sysutils;

const

         max_units = 20;
         max_order = 10;
         max_curve_points = 10;
         max_total_segments = 200;
         total_gen_tolerance = 0.01 ;
         ihr_tolerance = 0.000001 ;


         alpha : real = 0.5; {See note in loss matrix procedure}

type

         unit_array_real = array[1..max_units] of real;
         unit_name_array = array[1..max_units] of string;
         coefficients = array[0..max_order] of real;
         unit_poly_array = array[1..max_units] of coefficients;
         curve_points = array[0..max_curve_points] of real;
         unit_curve_array = array[1..max_units] of curve_points;
         system_ihr_array_real = array[1..max_total_segments] of real;
         system_ihr_array_integer = array[1..max_total_segments] of integer;
         B_matrix = array[1..max_units] of unit_array_real;
         filename_array = string;
         curvetype_list = (poly,pinc,pio);
         losstype_list = (noloss,constpf,lossform);
         solution_type_list = (lamsearch,tbllookup);
         schedtype_list = (totgen,totload);

var
        ioerr : boolean;
         ioval : integer;
     genname:unit_name_array;           {Generator name identifier}
     p:unit_array_real;                 {Present value of P}
     pmin:unit_array_real;              {Minimum MW}
     pmax:unit_array_real;              {Maximum MW}
     UR:unit_array_real;
     DR:unit_array_real;
     unitsebelum:unit_array_real;
     Ramprate:boolean;
     minihr:unit_array_real;            {Minimum unit incremental heat rate}
     maxihr:unit_array_real;            {Maximum unit incremental heat rate}
     fuelcost:unit_array_real;          {Fuel cost ( $/fuel unit )}
     coeff : unit_poly_array;           {Unit polynomial coefficients}
     ihr_mwpoint:unit_curve_array;       {MW points on ihr cost curve}
     ihr_cost:unit_curve_array;         {Cost points for ihr curve}
     io_mwpoint : unit_curve_array;     {MW Points on unit io curve}
     io_cost : unit_curve_array;        {Cost points on io curve}
     minput:unit_array_real;            {Minimum input for PINC curves }
     penfac:unit_array_real;            {Loss penalty factor}
     penfacD:unit_array_real;           {Loss penalty factor}
     unitmax:unit_array_real;           {maksimum unit generation awal}
     unitmin:unit_array_real;           {minimum unit generation awal}

     b00:real;                          {Loss matrix constant}
     b0:unit_array_real;                {Loss matrix linear terms}
     b:B_matrix;                        {Loss matrix quadratic terms}
     seginccost:system_ihr_array_real;  {Segment inc cost for table look up }
     segunit:system_ihr_array_integer;  {Unit associated with segment}
     segmw:system_ihr_array_real;       {MW contributed by segment}
     order:system_ihr_array_integer;    {Order routine output}
     ordvalue:system_ihr_array_real;    {Numbers to be ordered}
     inputfile:text;
     filename:filename_array;
     title1, title2 : string[80];
     print_output, diagflag, read_data : boolean;
     inputchar,quitflag : char;
     linenumber, ngen : integer;
     mwlosses, schedmw, lambda : real;
     curvetype_input, losstype_input : string[8];
     number_string : string[20];
     curveorder : integer;
     curvetype : curvetype_list;
     losstype : losstype_list;
     solution_type : solution_type_list;
     schedtype : schedtype_list;
     pgenmax, pgenmin, SRGenTotal : real;
     FF:Text;
     NoGenerator:integer ;
     NomerOrder:integer;
     ModeDataPembangkit:integer;
     ProsesRun :boolean;
     SR : real;
```

```
     IndividuSR:unit_array_real;

procedure datainput(namafile :string);
procedure ihr_ftn(i : integer; unitmw : real;  var unitihr : real );
procedure GetFileName(s :string;var d:string;var f:string;var e :string);
procedure datadump( var outfile:text );
procedure lambda_search_dispatch( var lambda : real );
procedure loss_matrix_ftn;
procedure inverse_ihr_ftn(i : integer; unitihr : real; var unitmw : real );

procedure table_lookup_dispatch( var lambda : real );
procedure order_routine(        numorder : integer;
                               ordertable : system_ihr_array_real;
                        var    orderindex : system_ihr_array_integer );
procedure output_routine( var outfile : text;
                               lambda : real );
procedure prod_cost(       i : integer;
                         unitmw : real;
              var    unitcost : real );
procedure dataOutput(namafile :string);
//procedure TotalGeneration (pmin,pmax,
function cekIoPoint(var unitG,Order:integer):boolean;
function cekIhrPoint(var unitG,Order:integer):boolean;
function CekEdcFile(filename:string): boolean;


implementation
function CekEdcFile(filename:string): boolean;
label keluar;
var s,d,f,e:string;
ff:text;
bc : boolean;
begin
bc:=false;
getfilename(filename,d,f,e);
 s:=trim(uppercase(f))+'.'+uppercase(e);
if s = 'EDC1.DAT' then bc:= true;
if s = 'EDCTEST.DAT' then bc:= true;

if s = 'EDC2.DAT' then bc:= true;
if s = 'EDC3.DAT'  then bc:= true;
if s = 'EDC4.DAT'  then bc:= true;
if s = 'EDC5.DAT'  then bc:= true;

if s = 'EX3A.DAT'  then bc:= true;
if s = 'EX3B.DAT'  then bc:= true;
if s = 'EX3C.DAT'  then bc:= true;
if s = 'EX3D.DAT'  then bc:= true;

if s = 'PR32.DAT'  then bc:= true;
if s = 'PR33.DAT'  then bc:= true;
if s = 'PR38.DAT'   then bc:= true;
if s = 'PR43A.DAT'  then bc:= true;
if s = 'PR43B.DAT'   then bc:= true;
if s = 'EX4D.DAT'   then bc:= true;

if bc = true then goto Keluar;

assign(ff,filename);
reset(ff);
readln(ff,s);
if pos('EDC FILE >>',s)>0 then bc:=true;
close(ff);


keluar:
cekedcfile:=bc;
end;
function cekIoPoint(var unitG,Order:integer):boolean;
label keluar;
var i,j:integer;
begin
  cekIopoint:=true;
  for i:=1 to ngen do
  begin
    for j:= 0 to curveorder-1 do begin
      if (io_mwpoint[i,j+1] - io_mwpoint[i,j])<=0 then
      begin
        cekIoPoint:=false;
        UnitG:=i;
        Order:=j;
        goto keluar;
      end;
    end;
  end;
keluar:
end;


function cekIhrPoint(var unitG,Order:integer):boolean;
```

```
label keluar;
var i,j:integer;
begin
  cekIhrpoint:=true;
  for i:=1 to ngen do
  begin
    for j:= 0 to curveorder-1 do begin
      if (ihr_mwpoint[i,j+1] - ihr_mwpoint[i,j])<=0 then
      begin
        cekIhrPoint:=false;
        UnitG:=i;
        Order:=j;
        goto keluar;
      end;
    end;
  end;
keluar:
end;

procedure prod_cost(       i : integer;
                      unitmw : real;
              var   unitcost : real );

      { Routine to return unit production cost given unit output in mw}
      { input : unit index = i}
      {         unit MW =  unitmw}
      { output: unit production cost = unitcost}

var
       j : integer;
       partmw, unitihr, ihr_a,ihr_b : real;

label return;

begin

case curvetype of
  poly :                          {Polynomial I/O curve}
      begin
      unitcost := 0;
      for j := curveorder downto 1 do
      unitcost := ( unitcost + coeff[ i,j ] ) * unitmw;

      unitcost := unitcost + coeff[ i,0 ];
      unitcost := unitcost * fuelcost[ i ];
      goto return
      end;

  pinc :                          {Piecewize incremental curve}
      begin
      j := 0;
      repeat
      j := j + 1;
      until (ihr_mwpoint[ i,j ] > unitmw) or (j = curveorder);

      ihr_a:=0.5*(ihr_cost[i,j]-ihr_cost[i,j-1])/(ihr_mwpoint[i,j]-ihr_mwpoint[i,j-
1]);
      ihr_b:=((ihr_mwpoint[i,j]*ihr_cost[i,j-1])-(ihr_mwpoint[i,j-
1]*ihr_cost[i,j]))/(ihr_mwpoint[i,j]-ihr_mwpoint[i,j-1]);

      unitcost:= ihr_a*unitmw*unitmw + ihr_b*unitmw + minput[i];
      unitcost:= unitcost*fuelcost[i];
      end;

   pio :                           {Piecewize I/O curve}
      begin
      for j := 1 to curveorder do
         begin
         if io_mwpoint[i,j] > unitmw then
            begin
            partmw := (unitmw-io_mwpoint[i,j-1] ) /
                                 (io_mwpoint[i,j]-io_mwpoint[i,j-1] );
            unitcost := io_cost[i,j-1] +
                              ( io_cost[i,j]-io_cost[i,j-1] ) * partmw;
            unitcost := unitcost * fuelcost[ i ];
            goto return
            end;
         if j = curveorder then     {Unit is at or above pmax}
            begin
            unitcost := io_cost[ i, j ] * fuelcost[ i ];
            goto return
            end;
          end;
      end;
  end;  { End of case statement}

return:

end; { End procedure }
```

```
procedure output_routine( var outfile : text;
                              lambda : real );

var
        limittxt : string[5];
        totalgen, totalcost, totalload : real;
        unitihr, unitinccost, unitcost : real;
        i : integer;

label return;

begin

writeln(outfile);
writeln(outfile,
     'generator  output   limit  inc cost penalty fact  operating cost');
writeln(outfile,
     '             mw              $/mwhr                    $/hr       ');
writeln(outfile,
     '---------  ------   -----  -------- ------------  --------------');

totalgen := 0.0;
totalcost := 0.0;

for i := 1 to ngen do
   begin
   write(outfile,genname[i]:9);
   write(outfile, ' ',p[i]:6:1, '    ');
   limittxt := '     ';
   if abs( p[i] - pmin[i] ) < total_gen_tolerance then limittxt := 'min  ';
   if abs( p[i] - pmax[i] ) < total_gen_tolerance then limittxt := 'max  ';
   write(outfile, limittxt );

   ihr_ftn( i, p[ i ], unitihr) ;    {Get unit incremental heat rate}

   unitinccost := unitihr * fuelcost[i];
   write(outfile, unitinccost:9:4);
   write(outfile, '    ',penfac[i]:9:4, '    ');

   prod_cost( i, p[ i ], unitcost );          {Calculate unit operating cost}

   writeln(outfile, '   ',unitcost:9:2);
   totalgen := totalgen + p[i];
   totalcost := totalcost + unitcost;
   end;
writeln(outfile,
     '---------  ------                                        --------------');
write(outfile, ' totals');
write(outfile, totalgen:9:1,
'                                  ', totalcost:9:2);
writeln(outfile);
writeln(outfile, ' lambda = ', lambda:10:4 );
writeln(outfile);

if (schedtype = totgen ) and ( losstype <> lossform ) then goto return;

if schedtype=totload then totalload := schedmw;

if schedtype=totgen then  totalload := totalgen - mwlosses;

writeln(outfile, 'total load = ',totalload:10:1,
                 '  total losses = ',mwlosses:10:1);

return:

end; { End procedure }

procedure order_routine(        numorder : integer;
                               ordertable : system_ihr_array_real;
                         var   orderindex : system_ihr_array_integer );

{ subroutine to order a list, least first                               }
{                                                                       }
{ input numorder = the number of items to be ordered                    }
{ input ordertable = the items to be ordered                            }
{ output orderindex = pointer to order value table                      }
{                                                                       }
{ nxt = Table used in order subroutine                                  }
{                                                                       }
  var
      stop:boolean;
      i,j,top,last,indx:integer;
      nxt : system_ihr_array_integer;

  begin
    for i := 1 to numorder do begin
      if (i <= 1) then begin
        top := 1;
        nxt[ 1 ] := 0;
        end
```

```
      else begin
        j := top;
        last := 0;
        repeat
          stop := true;
          if (ordertable[ i ] > ordertable[ j ]) then begin
            last := j;
            j := nxt[ j ];
            stop  := (j = 0);
            if (stop) then begin
              nxt[ last ] := i;
              nxt[ i ] := 0;
            end
            end
          else begin
            if (j <> top) then begin
              nxt[ last ] := i;             { j not = top }
              nxt[ i ] := j;
              end
            else begin
              top := i;                     { j = top }
              nxt[ i ] := j;
            end;
          end;
        until stop;
      end;
    end;
    indx := 1;
    j := top;
    repeat
      orderindex[ indx ] := j;
      j := nxt[ j ];
      indx := indx + 1;
    until (j = 0);
  end;

procedure table_lookup_dispatch( var lambda : real );

      {  Routine to perform economic dispatch by table look up}

var
         targetgen, ptotal, segihr, unitihr : real;
         i, j, k, kseg, kunit, numsegments : integer;
         done : boolean;

label return;

begin

if curvetype <> pio then
      begin
        { tmsg:=' ERROR -- must have piecewize i/o curves to use table lookup ';
         }
        {form5.showmodal;}
      goto return
      end;

if losstype = lossform then loss_matrix_ftn;  { Calc losses and pen factors}

if schedtype = totgen then targetgen := schedmw;
if schedtype = totload then targetgen := schedmw + mwlosses;

kseg := 0;                          {Build segment tables}
for i := 1 to ngen do
   begin
   for j := 1 to curveorder do
      begin
      kseg := kseg + 1;
      segihr := (io_cost[i,j]-io_cost[i,j-1]) /
                        (io_mwpoint[i,j]-io_mwpoint[i,j-1]);
      seginccost[ kseg ] := segihr * fuelcost[ i ] * penfac[ i ];
      segunit[ kseg ] := i;
      segmw[ kseg ] := io_mwpoint[i,j] - io_mwpoint[i,j-1];
      end;
   end;
numsegments := kseg;

                  {Set up for ordering routine}

for k := 1 to numsegments do
       ordvalue[k] := seginccost[k];
order_routine( numsegments, ordvalue, order );    {Call ordering routine}
                                                   {Result in order table}

             {Print segments table in incremental cost order}


  writeln(ff);
  writeln(ff,' segment number   inc cost      MW       unit ');
  writeln(ff,' --------------   ----------   ------    ----');
```

```
for k := 1 to numsegments do
begin
kseg := order[ k ];

   writeln(ff,'       ',kseg:3,'       ',seginccost[kseg]:10:4
                  ,segmw[kseg]:10:1,'   ',segunit[kseg]:4)

end;


      ptotal := 0.0;
      for i := 1 to ngen do
         begin
         p[ i ] := pmin[ i ];
         ptotal := ptotal + p[ i ]
         end;

      done := false;
      k := 0;
      repeat
         k := k + 1;
         kseg := order[ k ];
         kunit := segunit[ kseg ];
         if ( ptotal + segmw[ kseg ] ) < targetgen then
            begin
            p[ kunit ] := p[ kunit ] + segmw[ kseg ];
            ptotal := ptotal + segmw[ kseg ]
            end
            else
            begin
            p[ kunit ] := p[ kunit ] + ( targetgen - ptotal );
            done := true
            end;
      until done;

      lambda := seginccost[ kseg ];

return:

end; { End procedure }

procedure inverse_ihr_ftn(       i : integer;
                         unitihr : real;
                     var unitmw : real );

      { Routine to return unit MW given unit incremental heat rate}
      { input : unit number = i}
      {         unit inc heat rate = unitihr}
      { output: unit MW stored in unitmw}

label  return;

var
         unitihr1, delihr, partihr, dihrdp : real;
         segmentihr : real;
         j, step : integer;

begin

if unitihr >= maxihr[ i ] then
    begin
    unitmw := pmax[ i ];
    goto return
    end;
if unitihr <= minihr[ i ] then
    begin
    unitmw := pmin[ i ];
    goto return
    end;

case curvetype of
  poly :                         {Polynomial curve}
      begin
      if curveorder <= 1 then
       begin
       if unitihr > coeff[ i,1 ] then unitmw:=pmax[i] else unitmw:=pmin[i];
       goto return
       end;

      if curveorder = 2 then
       begin
       unitmw := ( unitihr - coeff[ i,1 ] ) / ( 2.0 * coeff[ i,2 ] );
       goto return
       end;

            { for curves of order >= 3 search for unitmw using Newtons method }

      unitmw := ( pmin[ i ] + pmax[ i ] )/ 2.0;
      step := 0;
```

```
      repeat
      step := step + 1;

      unitihr1 := 0;                    {Calc unitihr at unitmw as unitihr1}
      for j := curveorder downto 2 do
         unitihr1 := ( unitihr1 + j * coeff[i,j] ) * unitmw;
      unitihr1 := unitihr1 + coeff[i,1];
      delihr := unitihr - unitihr1;
      if abs( delihr ) < ihr_tolerance then goto return;

      dihrdp := 0;                       {Calc curve second derivative}
      for j := curveorder downto 3 do
         dihrdp := ( dihrdp + j*(j-1) * coeff[i,j] ) * unitmw;
      dihrdp := dihrdp + 2.0 * coeff[ i,2 ];
      unitmw := unitmw + delihr/dihrdp;

      until step > 20;

      goto return
      end;

  pinc :                             {Piecewize incremental curve}
      begin
      j := 0 ;
      repeat
      j := j + 1;
      until (ihr_cost[i,j] > unitihr) or
            (j = curveorder);

      partihr := ( unitihr - ihr_cost[i,j-1] )/
                          ( ihr_cost[i,j] - ihr_cost[i,j-1] );
      unitmw := ihr_mwpoint[i,j-1] +
                    ( ihr_mwpoint[i,j] - ihr_mwpoint[i,j-1] ) * partihr;
      goto return
      end;

   pio :                             {Piecewise I/O curve}
      begin
      j := 0;
      repeat
      j := j + 1;
      if j = curveorder then
         begin
         unitmw := io_mwpoint[i,j];
         goto return
         end;
      segmentihr :=(io_cost[i,j] - io_cost[i,j-1]) /
                           (io_mwpoint[i,j] - io_mwpoint[i,j-1]);
      until segmentihr >= unitihr;

      unitmw := io_mwpoint[ i,j-1 ];
      goto return
      end;

  end;  { End of case statement}

return:

end; { End procedure }

procedure loss_matrix_ftn;

      { Routine to calculate losses and penalty factors from loss formula}
      {  Input:   Table of unit generation p[ i ]}
      {           Loss formula b( i,j ), b0[ i ], b00}
      {  Output:  losses mwlosses and penalty factors penfac[ i ]}
      {  Note: loss formula expects p(i)'s to be in per unit so divide by 100.}

var
         i, j : integer;
         incloss, penfac_old, penfac_new : real;

label return;

begin

      mwlosses := b00;
      for i := 1 to ngen do
         begin
         mwlosses := mwlosses + b0[i] *
                          ( p[i]/100.0) + b[i,i] * sqr( p[i]/100.0 );
         for j := i+1 to ngen do
           mwlosses := mwlosses + 2.0 * ( p[i]/100.0) * ( p[j]/100.0 ) * b[i,j]
         end;

      mwlosses := mwlosses * 100.0;


      for i := 1 to ngen do
         begin
```

```
         penfac_old := penfac[ i ];
         incloss := b0[i];
         for j := 1 to ngen do
            incloss := incloss + 2.0 * ( p[j]/100.0 ) * b[i,j];
         penfac_new := 1.0 / ( 1.0 - incloss );
         penfac[ i ] := penfac_old + alpha *
                                      ( penfac_new  -  penfac_old)
         end;

{ Note, in the formula above the penalty factor is "filtered" by the }
{ alpha filtering constant. If alpha is set to 1.0 no filtering action }
{ takes place, if alpha is 0.0 penfac is constant at 1.0 , suggested }
{ value for alpha is 0.5 to 0.9 }

return:

end; { End procedure }


procedure lambda_search_dispatch( var lambda : real );

var
         i, n, lossiter : integer;
         lambdamin, lambdamax : real;
         lambdastart, deltalambda, targetgen : real;
         unitihr, unitmw, totalgen : real;
         endloop : boolean;

begin

for i := 1 to ngen do                    { Set unit output to midrange}
  begin
  p[ i ] := ( pmin[ i ] + pmax[ i ] ) / 2.0
  end;

lossiter := 0;
endloop := false;

repeat                                    {Top of iterative loop with losses}

  lambdamin := 10000.0;
  lambdamax := 0.0;
  mwlosses := 0 ;
  if losstype = lossform then   { Calc losses and pen factors}
  begin
    loss_matrix_ftn;
    writeln(ff);
    writeln(ff,' mw losses = ',mwlosses:10:1);
  end;

  for i := 1 to ngen do                  {Calculate max and min lambdas}
    begin
    ihr_ftn (i,pmax[i],maxihr[i]);
    lambda := maxihr[ i ] * penfac[ i ] * fuelcost[ i ];
    if lambda > lambdamax then lambdamax := lambda;
    ihr_ftn (i,pmin[i],minihr[i]);
    lambda := minihr[ i ] * penfac[ i ] * fuelcost[ i ];
    if lambda < lambdamin then lambdamin := lambda
    end;


    writeln(ff,' lambda limits = ',lambdamin:10:4,lambdamax:10:4);


  lambdastart := ( lambdamax + lambdamin ) / 2.0;
  deltalambda := ( lambdamax - lambdamin ) / 2.0;


   writeln(ff,' lambdastart deltalambda = ',lambdastart:10:4,deltalambda:10:4);
  {Set up total generation target}
  if schedtype = totgen then targetgen := schedmw;
  if schedtype = totload then targetgen := schedmw + mwlosses;
  {Lambda search}
  lambda := lambdastart;
  writeln(ff, ' targetgen = ',targetgen:10:1);

  n := 0;
  repeat                              {Top of lambda search loop}
      n := n + 1;
      totalgen := 0;
      for i := 1 to ngen do
         begin
         unitihr := lambda / ( penfac[ i ] * fuelcost[ i ] )  ;
         inverse_ihr_ftn( i, unitihr, unitmw );  {For given unitihr get unitmw}
         p[ i ] := unitmw;
         totalgen := totalgen + p[ i ]
         end;
```

```
            writeln(ff,' lambda = ',lambda:10:4,'  totalgen = ',totalgen:10:3);


       if abs( totalgen - targetgen ) >= total_gen_tolerance then
         begin
         if totalgen > targetgen then lambda := lambda - deltalambda;
         if totalgen < targetgen then lambda := lambda + deltalambda;
         deltalambda := deltalambda / 2.0
         end;

       until ( abs( totalgen - targetgen ) < total_gen_tolerance ) or
             ( n > 20 ) ;

{See if another loss iteration is needed}

       if losstype <> lossform then endloop := true;
       lossiter := lossiter + 1;
       if lossiter > 10 then endloop := true

until endloop;

end; { End Lambda search procedure }

procedure GetFileName(s :string;var d:string;var f:string;var e :string);
var ss ,sd:string;
  n:integer;
begin
  ss := s;
  d := '';
  n :=  pos('\',ss);
  while  n>0 do
  begin
    sd := copy(ss,1,n);
    d := d+sd;
    delete(ss,1,n);
  n :=  pos('\',ss);
  end;
  n := pos('.',ss);
  if n = 0 then
  begin
     f := ss;
     e:= '';
  end else
  begin
     f:=copy(ss,1,n-1);
     delete(ss,1,n);
     e := ss;
  end;
end;
procedure IOCheck( linenumber : integer );
begin
  IOVal := IOresult;
  IOErr := (IOVal <> 0);
end; { of proc IOCheck }


procedure datainput(namafile :string);

label quit;

var i,j,k,jj : integer;
    a : char;

begin
print_output := True;

linenumber := 0;

assign(inputfile, namafile);
     iocheck( linenumber );
     if ioerr then goto quit;
{                    }
{ Open data file     }
{                    }
reset(inputfile);
     iocheck( linenumber );
     if ioerr then goto quit;
{                    }
{ Read file header }
{                    }
linenumber := linenumber + 1;
readln( inputfile, title1 );
     iocheck( linenumber );
     if ioerr then goto quit;
linenumber := linenumber + 1;
readln( inputfile, title2 );
     iocheck( linenumber );
     if ioerr then goto quit;
```

```
     {                                                    }
     {  Read Number of generators, curve type, loss type  }
     {                                                    }
linenumber := linenumber + 1;
read( inputfile, ngen );
     iocheck( linenumber );
     if ioerr then goto quit;

repeat
read( inputfile, inputchar );
       iocheck( linenumber );
       if ioerr then goto quit;
until inputchar <> ' ';
curvetype_input := inputchar;
repeat
read( inputfile, inputchar );
      iocheck( linenumber );
      if ioerr then goto quit;
if inputchar <>' ' then curvetype_input := curvetype_input + inputchar;
until inputchar = ' ';

read( inputfile, curveorder );
      iocheck( linenumber );
      if ioerr then goto quit;

repeat
read(inputfile, inputchar );
     iocheck( linenumber );
     if ioerr then goto quit;
until inputchar <> ' ';
losstype_input := inputchar;
readln(inputfile);

 {                                                    }
 {  Set up internal variables for curvetype and losstype  }
 {                                                    }

if (curvetype_input = 'poly') or
   (curvetype_input = 'POLY') then curvetype := poly;

if (curvetype_input = 'pinc') or
   (curvetype_input = 'PINC') then curvetype := pinc;

if (curvetype_input = 'pio' ) or
   (curvetype_input = 'PIO' ) then curvetype := pio;

if (losstype_input = 'n'  ) or
   (losstype_input = 'N'  ) then losstype := noloss;

if (losstype_input = 'c'  ) or
   (losstype_input = 'C'  ) then losstype := constpf;

if (losstype_input = 'l'  ) or
   (losstype_input = 'L'  ) then losstype := lossform;


         {                     }
         { Read generator data }
         {                     }

for i := 1 to ngen do
   begin  { Read generator name }
      linenumber := linenumber + 1;
      repeat
      read( inputfile, inputchar );
            iocheck( linenumber );
            if ioerr then goto quit;
      until inputchar <> ' ';
      genname[i] := inputchar;
      repeat
      read( inputfile, inputchar );
            iocheck( linenumber );
            if ioerr then goto quit;
      if inputchar <>' ' then genname[i] := genname[i] + inputchar;
      until inputchar = ' ';
         {                                  }
         { Read generator min, max, fuelcost }
         {                                  }
      readln( inputfile, pmin[i], pmax[i], fuelcost[i], DR[i], UR[i], unitsebelum[i]
);
            iocheck( linenumber );
            if ioerr then goto quit;
         {                                  }
         { Read generator cost curve data    }
         {                                  }
      case curvetype of
        poly :
               begin  { read polynomial curve data}
               for j := 0 to curveorder do
                 begin
```

```
                    linenumber := linenumber + 1;
                    readln( inputfile, coeff[i,j] );
                     iocheck( linenumber );
                     if ioerr then goto quit;
                  end;
                end;

        pinc :
                begin  { read piecewise incremental cost curve data}
                linenumber := linenumber + 1;
                readln( inputfile, minput[i] );
                  iocheck( linenumber );
                  if ioerr then goto quit;
                for j := 0 to curveorder  do
                  begin
                    linenumber := linenumber + 1;
                    readln( inputfile, ihr_mwpoint[i,j], ihr_cost[i,j] );
                      iocheck( linenumber );
                      if ioerr then goto quit;
                  end;
                end;

        pio  :
                begin  { read piecewise I/O curve data}
                for j := 0 to curveorder do
                  begin
                    linenumber := linenumber + 1;
                    readln( inputfile, io_mwpoint[i,j], io_cost[i,j] );
                      iocheck( linenumber );
                      if ioerr then goto quit;
                  end;
                end;

        end;  { End of case statement}
    end;
          {                    }
          { Read loss data }
          {                    }

case losstype of

  noloss :
        begin
        for i := 1 to ngen do      { Init penalty factors}
              penfac[ i ] := 1.0
        end;

  constpf  :
       begin   { read constant penalty factor data}
         linenumber := linenumber + 1;
         for i := 1 to  ngen do
          begin
          if i < ngen then
          read( inputfile, penfac[i] )
          else
          readln( inputfile, penfac[ngen] );
             iocheck( linenumber );
             if ioerr then goto quit
          end;
       end;

  lossform :

       begin  { read loss formula data}

       linenumber := linenumber + 1;
       readln( inputfile, b00 );
             iocheck( linenumber );
             if ioerr then goto quit;

       linenumber := linenumber + 1;
       for j := 1 to  ngen  do
         begin
         if j < ngen then
            read( inputfile, b0[ j ] )
            else
            readln( inputfile, b0[ngen] );
             iocheck( linenumber );
             if ioerr then goto quit;
         end;

       for i := 1 to ngen do
         begin
         linenumber := linenumber + 1;
         for j := 1 to ngen  do
          begin
          if j < ngen then
             read( inputfile, b[ i, j ] )
             else
             readln( inputfile, b[ i, ngen ] );
```

```
            iocheck( linenumber );
            if ioerr then goto quit
         end;
        end;
        for i := 1 to ngen do      { Init penalty factors}
            penfac[ i ] := 1.0
      end;
  end;  { End of case statement}
{                                  }
{  End of data input, close file   }
{                                  }

close ( inputfile );

{A very useful table is the incremental cost at the max and min of each unit. }
{These are calculated and stored in tables maxihr and minihr.}
{Also calculate the max and min generation}



quit:

end;   { end of data input }




procedure dataOutput(namafile :string);
var i,j,k,jj : integer;
    a : char;
    d,e,f:string;
   var inputfile:text;
begin
getfilename(namafile,d,f,e);
assign(inputfile, namafile);
rewrite(inputfile);
writeLn( inputfile, 'EDC FILE >> '+f+'.'+e );
writeln( inputfile, '====================================');
write( inputfile, ngen );
write(inputfile,' ');

if  curvetype = poly then write(inputfile,'POLY');
if  curvetype = pinc then write(inputfile,'PINC');
if  curvetype = PIO then write(inputfile,'PIO');
write(inputfile,' ');
write(inputfile,curveorder);
write(inputfile,' ');

if losstype = noloss  then write(inputfile,'NOLOSS');
if losstype = constpf  then write(inputfile,'CONSTPF');
if losstype = lossform  then write(inputfile,'LOSSFORM');
writeln(inputfile);
for i := 1 to ngen do
   begin
      write(inputfile,genname[i]);
      write(inputfile,' ');
      writeln( inputfile, pmin[i]:15:6, pmax[i]:15:6, fuelcost[i]:15:6, DR[i]:15:6,
UR[i]:15:6, unitsebelum[i]:15:6 );
    case curvetype of
        poly :
               begin
               for j := 0 to curveorder do
                 begin
                   writeln( inputfile, coeff[i,j]:15:6 );
                 end;
               end;

        pinc :
               begin
               writeln( inputfile, minput[i]:15:6 );
               for j := 0 to curveorder  do
                 begin
                   writeln( inputfile, ihr_mwpoint[i,j]:15:6, ihr_cost[i,j]:15:6 );
                 end;
               end;

        pio  :
               begin
               for j := 0 to curveorder do
                 begin
                   writeln( inputfile, io_mwpoint[i,j]:15:6, io_cost[i,j]:15:6);
                 end;
               end;

        end;
    end;

case losstype of
```

```
  noloss :
        begin
        for i := 1 to ngen do      { Init penalty factors}
              penfac[ i ] := 1.0
        end;

  constpf  :
      begin
         for i := 1 to  ngen do
          begin
          if i < ngen then
          begin
          write( inputfile, penfac[i]);
          write(inputfile,' ');

          end
          else writeln( inputfile, penfac[ngen]);
          end;
      end;

  lossform :

      begin

      linenumber := linenumber + 1;
      writeln( inputfile, b00);
       for j := 1 to  ngen  do
        begin
        if j < ngen then
           begin
           write( inputfile, b0[ j ]);
           write(inputfile,' ');
           end
           else
           writeln( inputfile, b0[ngen]);
        end;

      for i := 1 to ngen do
        begin
        for j := 1 to ngen  do
         begin
         if j < ngen then
            begin
            write( inputfile, b[i, j]);
            write(inputfile,' ');

            end
            else
            writeln( inputfile, b[i,ngen]);
         end;
        end;

        for i := 1 to ngen do      { Init penalty factors}
              penfac[ i ] := 1.0

      end;

  end;


close ( inputfile );


end;

procedure ihr_ftn(          i : integer;
                    unitmw : real;
                var unitihr : real );

      { Routine to return unit incremental heat rate given unit output in MW }
      { input : unit index = i}
      {         unit mw = unitmw}
      { output: unit incremental heat rate = unitihr}

var
        partmw : real;
        j : integer;

begin

case curvetype of
  poly :                    {Polynomial I/O curve}
      begin
      unitihr := 0.0;
      for j := curveorder downto 2 do
        begin
        unitihr := ( unitihr + j * coeff[ i,j ] ) * unitmw;
        end;
      unitihr := unitihr + coeff[ i,1 ]
```

```
      end;

  pinc :                      {Piecewize incremental curve}
      begin
      j := 0;
      repeat
      j := j + 1;
      until (ihr_mwpoint[ i,j ] > unitmw) or (j = curveorder);

      partmw := (unitmw - ihr_mwpoint[i,j-1] )/
                        (ihr_mwpoint[i,j] - ihr_mwpoint[i,j-1] );
      unitihr := ihr_cost[i,j-1] + ( ihr_cost[i,j] - ihr_cost[i,j-1] ) * partmw
      end;

  pio  :                      {Piecewize I/O curve}
      begin
      j := 0;
      repeat
      j := j + 1;
      until (io_mwpoint[ i,j ] > unitmw) or (j = curveorder);

      unitihr := (io_cost[i,j] - io_cost[i,j-1] ) /
                          ( io_mwpoint[i,j] - io_mwpoint[i,j-1] )
      end;
  end;  { End of case statement}


end; { End ihr_ftn procedure }

procedure datadump( var outfile:text );


var
   i,j : integer;

begin
for i := 1 to ngen do
begin
pmax[i]:=unitmax[i]-(individuSR[i]/100)*unitmax[i];
pmin[i]:=unitmin[i];
if unitsebelum[i] <> 0 then if ramprate then
  begin
  if (unitsebelum[i]+UR[i])<(unitmax[i]-(IndividuSR[i]/100)*unitmax[i]) then pmax[i]
:= unitsebelum[i]+UR[i];
  if (unitsebelum[i]-DR[i])>(unitmin[i]) then pmin[i] := unitsebelum[i]-DR[i];
  end;
end;

writeln(outfile);
writeln(outfile, title1);
writeln(outfile, title2);
writeln(outfile);
writeln(outfile ,' number of generator units = ',ngen );

case curvetype of
  poly : writeln(outfile ,' unit curve type = poly ');
  pinc : writeln(outfile ,' unit curve type = pinc ');
  pio  : writeln(outfile ,' unit curve type = pio');
  end;  { End of case statement}

writeln(outfile ,' curve order = ',curveorder);

case losstype  of
  noloss :   writeln(outfile ,' network loss representation = noloss ');
  constpf :  writeln(outfile ,' network loss representation = constpf ');
  lossform : writeln(outfile ,' network loss representation = lossform ');
  end;  { End of case statement}

for i := 1 to ngen do
  begin
   writeln(outfile);
   write(outfile, genname[i],' limits = ',pmin[i]:7:2, ' ',pmax[i]:7:2 );
   writeln(outfile, ' fuelcost = ',fuelcost[i]:10:4 );
     case curvetype of
          poly :
              begin
              writeln(outfile,'  polynomial coefficients' );
              for j := 0 to curveorder do
                begin
                writeln(outfile, coeff[i,j]:15:6);
                end;
              end;
          pinc :
              begin
              writeln(outfile,' incremental cost curve points');
              writeln(outfile,'input at pmin = ',minput[i]:10:2);
              for j := 0 to curveorder do
                begin
                writeln(outfile,ihr_mwpoint[i,j]:9:2,ihr_cost[i,j]:9:3 )
                end;
```

```
                writeln(outfile);
                end;
            pio :
                begin
                writeln(outfile,' cost curve points');
                for j := 0 to curveorder do
                  begin
                  writeln(outfile,io_mwpoint[i,j]:9:2,'  ',io_cost[i,j]:9:3 )
                  end;
                writeln(outfile);
                end;
            end; {end of case statement }
  end;
case losstype  of

  constpf :
                begin
                writeln(outfile);
                writeln(outfile,' Penalty Factors');
                for i := 1 to ngen do
                  begin
                  writeln(outfile,' penalty factor ',i,' ',penfac[i]:10:3)
                  end;
                writeln(outfile);
                end;

  lossform :
                begin
                writeln(outfile);
                writeln(outfile,' Loss Formula');
                writeln(outfile,'B00 = ',b00:10:4);
                writeln(outfile);
                writeln(outfile,'B0 = ');
                for i := 1 to ngen do
                   if i < ngen then
                   write(outfile,b0[i]:10:4,' ')
                   else
                   writeln(outfile,b0[i]:10:4);
                writeln(outfile);
                writeln(outfile,' B = ');
                for i := 1 to ngen do
                   begin
                        for j := 1 to ngen do
                           if j < ngen then
                           write(outfile,b[i,j]:10:4,' ')
                           else
                           writeln(outfile,b[i,ngen]:10:4);
                   end;
                writeln(outfile)
                end;
  end;  { End of case statement}
    if solution_type = lamsearch then writeln(outfile,'using lambda search')
                                 else writeln(outfile,'using tablelookup');
       writeln(outfile);
       if schedtype = totgen then
          writeln(outfile, ' total generation schedule = ',schedmw:10:1)
       else
          writeln(outfile, ' total load schedule = ', schedmw:10:1);
       if losstype = lossform then writeln(outfile, ' using loss formula ')
                              else writeln(outfile,' losses neglected');
       writeln(outfile);
end; { End procedure }


end.
```

Unit 4

```
unit Unit4;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls,unit3;

type
  TFrmEDCSetUpSolution = class(TForm)
    GroupBox1: TGroupBox;
    RbLamSearch: TRadioButton;
    Rbtbllookup: TRadioButton;
    GroupBox2: TGroupBox;
    GroupBox3: TGroupBox;
    RbTotGen: TRadioButton;
    RbTotLoad: TRadioButton;
    Ed: TEdit;
    Lb: TLabel;
    LbMaxGe: TLabel;
    LbMinG: TLabel;
    Button1: TButton;
```

```
    LbSpinningReserve: TLabel;
    GroupBox4: TGroupBox;
    RbRampYes: TRadioButton;
    RbRampNo: TRadioButton;
    procedure RbTotGenClick(Sender: TObject);
    procedure RbTotLoadClick(Sender: TObject);
    procedure RbLamSearchClick(Sender: TObject);
    procedure RbtbllookupClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FrmEDCSetUpSolution: TFrmEDCSetUpSolution;

implementation

uses Unit5;

{$R *.dfm}

procedure TFrmEDCSetUpSolution.RbTotGenClick(Sender: TObject);
begin
schedtype := totgen;
lb.caption :='Enter Total Generation';
end;

procedure TFrmEDCSetUpSolution.RbTotLoadClick(Sender: TObject);
begin
schedtype := totload;
lb.caption :='Enter Total Load';
end;

procedure TFrmEDCSetUpSolution.RbLamSearchClick(Sender: TObject);
begin
solution_type  := lamsearch;
end;

procedure TFrmEDCSetUpSolution.RbtbllookupClick(Sender: TObject);
begin
solution_type  := tbllookup;
end;

procedure TFrmEDCSetUpSolution.FormActivate(Sender: TObject);
var s:string;
i:integer;
begin
     pgenmax := 0.0;
pgenmin := 0.0;
SRGenTotal :=0.0;

for i := 1 to ngen do
  begin
    ihr_ftn( i, pmax[ i ], maxihr[ i ] );
    ihr_ftn( i, pmin[ i ], minihr[ i ] );

{ calculate maximum and minimum generation available from generators }
    SRGenTotal := SRgenTotal + ((IndividuSR[i]/100)*pmax[i]);
    pmax[i] := pmax[i] - ((IndividuSR[i]/100)*pmax[i]);
    pgenmax := pgenmax + pmax[ i ];
    pgenmin := pgenmin + pmin[ i ];

  end;

prosesrun:=false;
str(pgenmax:10:1,s);
lbMaxGe.caption:=' Maximum generation is :'+s;
str(pgenmin:10:1,s);
lbming.caption:= ' Minimum generation is :'+s;
str (SRGenTotal:10:1,s);
LbSpinningReserve.caption := ' Spinning Reserve is : '+s;
LbSpinningReserve.Visible:=true;
if SRGenTotal =0 then LbSpinningReserve.Visible:=false;
end;



 procedure PRoses;
var s,e,f,d:string;
 k:integer;
begin
    if FrmEDCSetUpSolution.RbRampYes.Checked then Ramprate:=true else
RampRate:=false;
    s:=frmEdcSetupSolution.ed.text;
```

```
    val(s, schedmw,k);
    if (schedmw<pgenmin) or (schedmw>pgenmax) then
    begin
      showmessage('DED not possible with that scheduled generation');
      prosesrun:=false;
    end
    else
    begin
 prosesrun:=true;
       getfilename(filename,d,f,e);
       title1 := 'File Name : '+f+'.'+e;
       assign(ff,'data.dum');
       rewrite(Ff);
       datadump(Ff) ;
       if solution_type = lamsearch  then lambda_search_dispatch( lambda );
       if solution_type = tbllookup  then table_lookup_dispatch( lambda );
       output_routine(ff, lambda ) ;
       close(ff);
       frmEdcSetupSolution.Close ;
    end;


 end;
procedure TFrmEDCSetUpSolution.Button1Click(Sender: TObject);
begin
 title1:='EDC FILE ' +fileName;


 proses;

end;


procedure TFrmEDCSetUpSolution.FormClose(Sender: TObject;
  var Action: TCloseAction);
var i:integer;
begin
for i :=1 to ngen do
begin
pmax[i]:=unitmax[i];
pmin[i]:=unitmin[i];
end;
end;

end.
```

Unit 5

```
unit Unit5;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, StdCtrls,unit3;

type
  TfrmEdcPenFac = class(TForm)
    sgPenFac: TStringGrid;
    BtOk: TButton;
    BtCancel: TButton;
    procedure FormActivate(Sender: TObject);
    procedure BtCancelClick(Sender: TObject);
    procedure BtOkClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmEdcPenFac: TfrmEdcPenFac;

implementation

{$R *.dfm}

procedure TfrmEdcPenFac.FormActivate(Sender: TObject);
var i:integer;
    s:string;
begin
sgpenfac.ColCount :=ngen+1;
for i:=1 to ngen do
begin
str(i,s);
sgpenfac.Cells[i,0]:=genname[i];
str(penfac[i]:15:5,s);
sgpenfac.Cells[i,1]:=trim(s);
```

```
end;

end;

procedure TfrmEdcPenFac.BtCancelClick(Sender: TObject);
begin
close;
end;

procedure TfrmEdcPenFac.BtOkClick(Sender: TObject);
var i,k:integer;
begin
for i:=1 to ngen do
begin
val(sgpenfac.Cells[i,1],penfac[i],k);

end;

close;
end;

end.
```

Unit 6

```
unit Unit6;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls,unit3;

type
  TFrmEdcResult = class(TForm)
    Memo1: TMemo;
    Button1: TButton;
    SdSave: TSaveDialog;
    BTSAVE: TButton;
    procedure Button1Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure BTSAVEClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FrmEdcResult: TFrmEdcResult;

implementation

{$R *.dfm}

procedure TFrmEdcResult.Button1Click(Sender: TObject);
begin
close;
end;

procedure Loaddum;
var s:string;
    f :text;
begin
 FrmEdcResult.memo1.Clear ;
 assign(f,'data.Dum');
 reset(f);
 frmedcresult.Memo1.Clear;
 while not(eof(f)) do
 begin
   readln(f,s);
   FrmEdcResult.memo1.Lines.Add(s);
 end;
 close(f);
end;

procedure TFrmEdcResult.FormActivate(Sender: TObject);
begin
loADDUM;
end;
procedure saveResult;
var ffx : text;
    fn:string;
    i:integer;
    e,f,d,ss:string;

begin
 getfilename(filename,d,f,e);
 frmedcresult.SdSave.DefaultExt :='txt';
 frmedcresult.SdSave.InitialDir :=d;
```

```
 frmedcresult.SdSave.FileName  :=trim(f)+'_Report';


if frmedcResult.SdSave.Execute then
begin

  fn:=  frmedcResult.SdSave.FileName ;
   ss:=filename;
  assign(ffx,fn);
  rewrite(ffx);

  for i:=0 to frmedcResult.memo1.Lines.Count -1 do
  begin
  writeln(ffx,frmedcResult.memo1.Lines.Strings[i]);
  end;
  close(FFX);

end;

end;


procedure TFrmEdcResult.BTSAVEClick(Sender: TObject);
begin
 saveresult;
end;



end.
```

Unit 7

```
unit Unit7;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls,unit3, Grids;

type
  TFrmEdcDataPembangkit = class(TForm)
    GroupBox1: TGroupBox;
    EdNamaUnit: TEdit;
    GroupBox2: TGroupBox;
    GroupBox3: TGroupBox;
    Label1: TLabel;
    EdMin: TEdit;
    EdMaximum: TEdit;
    Maximum: TLabel;
    GroupBox4: TGroupBox;
    EdFUelCost: TEdit;
    BtOk: TButton;
    BtCancel: TButton;
    SG: TStringGrid;
    GroupBox5: TGroupBox;
    EdDR: TEdit;
    EdUR: TEdit;
    EdMinInput: TEdit;
    edunitsebelum: TEdit;
    BtPrev: TButton;
    BtNext: TButton;
    edIndividuSR: TEdit;
    procedure FormActivate(Sender: TObject);
    procedure BtOkClick(Sender: TObject);
    procedure BtCancelClick(Sender: TObject);
    procedure BtPrevClick(Sender: TObject);
    procedure BtNextClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FrmEdcDataPembangkit: TFrmEdcDataPembangkit;

implementation

uses Unit1;
var
NnOrde:integer;
{$R *.dfm}
Procedure SaveInformation ;
var i,k,j:integer;
   s:string;
begin
 genname[nogenerator]:=trim(frmedcdatapembangkit.ednamaunit.Text);
 s:=frmedcdatapembangkit.edmaximum.Text;
```

```
 val(s,pmax[nogenerator],k);
 unitmax[nogenerator]:=pmax[nogenerator];
 s:=trim(frmedcdatapembangkit.edmin.Text);
 val(s,pmin[nogenerator],k);
 unitmin[nogenerator]:=pmin[nogenerator];
 s:=trim(frmedcdatapembangkit.edDR.Text);
 val(s,DR[nogenerator],k);
 s:=trim(frmedcdatapembangkit.edUR.Text);
 val(s,UR[nogenerator],k);
 s:=frmedcdatapembangkit.edFuelCost.Text;
 val(s,fuelcost[nogenerator],k);
 s:=frmedcdatapembangkit.edunitsebelum.Text;
 val(s,unitsebelum[nogenerator],k);
 s:=frmedcdatapembangkit.edIndividuSR.Text;
 val(s,IndividuSR[nogenerator],k);

   frmEdcMain.sgNamaPembangkit.rowcount:=ngen+1;

 for i:=1 to ngen do
 begin
   str(i,s);
   frmEdcMain.sgnamapembangkit.cells[0,i]:= s;
   frmEdcMain.sgnamapembangkit.cells[1,i]:= genname[i] ;
   str(pmin[i]:7:2,s);
   frmEdcMain.sgnamapembangkit.cells[2,i]:= trim(s);
   str(pmax[i]:7:2,s);
   frmEdcMain.sgnamapembangkit.cells[3,i]:= trim(s);
   str(fuelcost[i]:6:3,s);
   frmEdcMain.sgnamapembangkit.cells[4,i]:= trim(s);
 end;
 if curvetype =poly then
 begin
  for j:=0 to curveorder do
  begin
  val(trim(frmedcdatapembangkit.sg.cells[1,1+j]),coeff[nogenerator,j],k);
  end;
end;

if curvetype =  pio  then
begin
  for j := 0 to curveorder do
  begin

    val(trim(frmedcdatapembangkit.sg.Cells[1,1+j]),io_mwpoint[nogenerator,j],k);
    val(trim(frmedcdatapembangkit.sg.Cells[2,1+j]),io_cost[nogenerator,j],k);
  end;
end;
if curvetype =  pinc  then
begin

  val(trim(frmedcdatapembangkit.edminInput.text),minput[nogenerator],k);
  for j := 0 to curveorder do
  begin
    val(trim(frmedcdatapembangkit.sg.Cells[1,1+j]),ihr_mwpoint[nogenerator,j],k);
    val(trim(  frmedcdatapembangkit.sg.Cells[2,1+j]),ihr_cost[nogenerator,j],k);
  end;
end;
end;



Procedure SetInformation;
var s,s1,s2,s3:string;
  j:integer;
begin
FrmedcDataPembangkit.ednamaunit.Text := genname[ NoGenerator];
str(pmax[nogenerator]:10:3,s);
FrmedcDataPembangkit.edmaximum.Text := trim(s);
str(pmin[nogenerator]:10:3,s);
FrmedcDataPembangkit.edmin.Text := trim(s);
str(fuelcost[nogenerator]:10:3,s);
FrmedcDataPembangkit.edFuelCost.Text := trim(s);
str(DR[nogenerator]:10:3,s);
frmedcDataPembangkit.edDR.Text := trim(s);
str(UR[nogenerator]:10:3,s);
frmedcDataPembangkit.edUR.Text := trim(s);
str(unitsebelum[nogenerator]:10:3,s);
frmedcDataPembangkit.edunitsebelum.Text := trim(s);
str(IndividuSR[nogenerator]:2:0,s);
frmedcDataPembangkit.edIndividuSR.Text := trim(s);
FrmedcDataPembangkit.groupbox5.Visible := false ;
if curvetype =poly then
begin
  FrmedcDataPembangkit.Groupbox2.Caption := 'Orde Polynomial';
  FrmedcDataPembangkit.sg.Cells[1,0]:='Nilai';
  FrmedcDataPembangkit.sg.Cells[0,0]:='Orde Ke';
  FrmedcDataPembangkit.sg.ColWidths[1] := 80;
  FrmedcDataPembangkit.sg.ColCount :=2;
  FrmedcDataPembangkit.sg.FixedRows :=1;
```

```
  FrmedcDataPembangkit.sg.RowCount :=curveorder+2;
  for j:=0 to curveorder do
  begin
  str(coeff[nogenerator,j]:15:6,s1);
  str(j:2,s2);

  FrmedcDataPembangkit.sg.Cells[1,1+j]:=s1;
  FrmedcDataPembangkit.sg.Cells[0,1+j]:=s2;

  end;
end;
{===}

if curvetype =  pio  then
begin
  FrmedcDataPembangkit.Groupbox2.Caption := 'Piecewise Input/Output Curve';
  FrmedcDataPembangkit.sg.ColCount :=3;
  FrmedcDataPembangkit.sg.Cells[2,0]:='Io Cost';
  FrmedcDataPembangkit.sg.Cells[1,0]:='P(MW)';
  FrmedcDataPembangkit.sg.Cells[0,0]:='Point';
  FrmedcDataPembangkit.sg.ColWidths[1] := 80;
  FrmedcDataPembangkit.sg.ColWidths[2] := 80;

  FrmedcDataPembangkit.sg.FixedRows :=1;
  FrmedcDataPembangkit.sg.RowCount :=curveorder+2;

  for j := 0 to curveorder do
  begin
    str( io_mwpoint[nogenerator,j]:15:6,s1);
    str(io_cost[nogenerator,j]:15:6,s2);
    str(j,s3);
    FrmedcDataPembangkit.sg.Cells[1,1+j]:=trim(s1);
    FrmedcDataPembangkit.sg.Cells[0,1+j]:=trim(s3);
    FrmedcDataPembangkit.sg.Cells[2,1+j]:=trim(s2);

  end;
end;
if curvetype =  pinc  then
begin
  FrmedcDataPembangkit.groupbox5.Visible := true ;
  str(minput[nogenerator]:15:6,s1);
  FrmedcDataPembangkit.edMinInput.text:=s1;
  FrmedcDataPembangkit.Groupbox2.Caption := 'Piecewise Incremental Heat Rate Curve';
  FrmedcDataPembangkit.sg.ColCount :=3;
  FrmedcDataPembangkit.sg.Cells[2,0]:='IHR';
  FrmedcDataPembangkit.sg.Cells[1,0]:='P(MW)';
  FrmedcDataPembangkit.sg.Cells[0,0]:='Point';
  FrmedcDataPembangkit.sg.ColWidths[1] := 80;
  FrmedcDataPembangkit.sg.ColWidths[2] := 80;

  FrmedcDataPembangkit.sg.FixedRows :=1;
  FrmedcDataPembangkit.sg.RowCount :=curveorder+2;

  for j := 0 to curveorder do
  begin
    str( ihr_mwpoint[nogenerator,j]:15:6,s1);
    str(ihr_cost[nogenerator,j]:15:6,s2);
    str(j,s3);
    FrmedcDataPembangkit.sg.Cells[1,1+j]:=trim(s1);
    FrmedcDataPembangkit.sg.Cells[0,1+j]:=trim(s3);
    FrmedcDataPembangkit.sg.Cells[2,1+j]:=trim(s2);
  end;
end;
{===}

end;

procedure TFrmEdcDataPembangkit.FormActivate(Sender: TObject);
begin
if modedatapembangkit = 0 then
begin
  btnext.Visible :=true;
  btPrev.visible :=true;
end;

if modedatapembangkit = 1 then
begin
 btnext.Visible :=false;
 btPrev.visible :=false;

end;

setinformation;
end;

procedure TFrmEdcDataPembangkit.BtOkClick(Sender: TObject);
var i,k,j:integer;
    s:string;
begin
if modedatapembangkit = 1 then
```

```
            begin
              ngen:=nogenerator;
            end;
            saveinformation;
            close;
            end;


            procedure TFrmEdcDataPembangkit.BtCancelClick(Sender: TObject);
            begin
            close;
            end;

            procedure TFrmEdcDataPembangkit.BtPrevClick(Sender: TObject);
            begin
            saveinformation;
             nogenerator:=nogenerator-1;
             if nogenerator<1 then nogenerator := 1;
              setinformation;
            end;

            procedure TFrmEdcDataPembangkit.BtNextClick(Sender: TObject);
            begin
            saveinformation;
            nogenerator:=nogenerator+1;
            if nogenerator>ngen then nogenerator := Ngen;
            setinformation;
            end;

end.
```

Hasil Perhitungan *Software*

Kasus 3

periode 1

generator	output mw	limit	inc cost $/mwhr	penalty fact	operating cost $/hr
UNIT1	15.0	max	40.5405	1.0000	558.21
UNIT2	45.0	max	76.7110	1.0000	2316.98
UNIT3	25.0	max	33.2410	1.0000	806.10
UNIT4	25.0	max	30.6445	1.0000	766.53
UNIT5	25.0	max	30.6445	1.0000	766.53
UNIT6	13.7		100.5353	1.0000	1917.66
UNIT7	13.7		100.5353	1.0000	1917.66
UNIT8	12.3	max	25.8302	1.0000	284.00
UNIT9	12.3	max	25.8302	1.0000	284.00
UNIT10	12.3	max	25.8302	1.0000	284.00
UNIT11	12.3	max	25.8302	1.0000	284.00
UNIT12	24.0	max	88.2493	1.0000	2314.47
UNIT13	6.4		100.5353	1.0000	1414.27
UNIT14	36.2	max	88.1572	1.0000	3277.25
UNIT15	45.0	max	94.5805	1.0000	3971.96
UNIT16	37.0	max	87.1034	1.0000	3214.70
UNIT17	45.0	max	93.8360	1.0000	3938.46
UNIT18	6.4		100.5353	1.0000	1414.27
totals	411.6				29731.07
lambda =	100.5353				

total load = 411.6 total losses = 0.0

periode 2

generator	output mw	limit	inc cost $/mwhr	penalty fact	operating cost $/hr
UNIT1	15.0	max	40.5405	1.0000	558.21
UNIT2	45.0	max	76.7110	1.0000	2316.98
UNIT3	25.0	max	33.2410	1.0000	806.10
UNIT4	25.0	max	30.6445	1.0000	766.53
UNIT5	25.0	max	30.6445	1.0000	766.53
UNIT6	8.2		92.4632	1.0000	1387.60
UNIT7	8.2		92.4632	1.0000	1387.60
UNIT8	12.3	max	25.8302	1.0000	284.00
UNIT9	12.3	max	25.8302	1.0000	284.00
UNIT10	12.3	max	25.8302	1.0000	284.00
UNIT11	12.3	max	25.8302	1.0000	284.00
UNIT12	24.0	max	88.2493	1.0000	2314.47
UNIT13	3.1		92.4632	1.0000	1099.29
UNIT14	36.2	max	88.1572	1.0000	3277.25
UNIT15	42.5		92.4632	1.0000	3736.68

```
   UNIT16   37.0    max    87.1034      1.0000          3214.70
   UNIT17   43.4           92.4632      1.0000          3786.52
   UNIT18    3.1           92.4632      1.0000          1099.29
---------  ------                                    -------------
 totals    389.9                                       27653.74
 lambda =   92.4632

total load =      389.9  total losses =        0.0
```

periode 3

```
generator  output   limit  inc cost penalty fact  operating cost
             mw             $/mwhr                     $/hr
---------  ------   -----  -------- ------------  -------------
    UNIT1   15.0    max    40.5405      1.0000          558.21
    UNIT2   45.0    max    76.7110      1.0000         2316.98
    UNIT3   25.0    max    33.2410      1.0000          806.10
    UNIT4   25.0    max    30.6445      1.0000          766.53
    UNIT5   25.0    max    30.6445      1.0000          766.53
    UNIT6    3.0    min    84.8020      1.0000          925.52
    UNIT7    3.0    min    84.8020      1.0000          925.52
    UNIT8   12.3    max    25.8302      1.0000          284.00
    UNIT9   12.3    max    25.8302      1.0000          284.00
   UNIT10   12.3    max    25.8302      1.0000          284.00
   UNIT11   12.3    max    25.8302      1.0000          284.00
   UNIT12   20.7           83.9472      1.0000         2032.61
   UNIT13    3.0    min    92.0946      1.0000         1085.53
   UNIT14   30.9           83.9472      1.0000         2818.17
   UNIT15   32.4           83.9472      1.0000         2844.12
   UNIT16   33.2           83.9472      1.0000         2893.95
   UNIT17   33.2           83.9472      1.0000         2893.95
   UNIT18    3.0    min    92.0946      1.0000         1085.53
---------  ------                                    -------------
 totals    346.6                                       23855.28
 lambda =   83.9472

total load =      346.6  total losses =        0.0
```

periode 4

```
generator  output   limit  inc cost penalty fact  operating cost
             mw             $/mwhr                     $/hr
---------  ------   -----  -------- ------------  -------------
    UNIT1   15.0    max    40.5405      1.0000          558.21
    UNIT2   44.6           76.2671      1.0000         2288.82
    UNIT3   25.0    max    33.2410      1.0000          806.10
    UNIT4   25.0    max    30.6445      1.0000          766.53
    UNIT5   25.0    max    30.6445      1.0000          766.53
    UNIT6    3.0    min    84.8020      1.0000          925.52
    UNIT7    3.0    min    84.8020      1.0000          925.52
    UNIT8   12.3    max    25.8302      1.0000          284.00
    UNIT9   12.3    max    25.8302      1.0000          284.00
   UNIT10   12.3    max    25.8302      1.0000          284.00
   UNIT11   12.3    max    25.8302      1.0000          284.00
   UNIT12   14.9           76.2671      1.0000         1564.46
   UNIT13    3.0    min    92.0946      1.0000         1085.53
   UNIT14   21.1           76.2671      1.0000         2038.55
   UNIT15   23.2           76.2671      1.0000         2113.07
   UNIT16   24.1           76.2671      1.0000         2162.91
   UNIT17   24.1           76.2671      1.0000         2162.91
   UNIT18    3.0    min    92.0946      1.0000         1085.53
---------  ------                                    -------------
 totals    303.3                                       20386.21
 lambda =   76.2671

total load =      303.3  total losses =        0.0
```

final output

```
FINAL OUTPUT DYNAMIC ECONOMIC DISPATCH

PERIOD       UNIT GENERATION
PCOST     LOAD
             1        2        3        4        5        6        7        8        9       10       11
12       13       14       15       16       17       18          R/HR       MW
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
   1      15.0     45.0     25.0     25.0     25.0     13.7     13.7     12.3     12.3     12.3     12.3
24.0      6.4     36.2     45.0     37.0     45.0      6.4   29731.07    411.6
   2      15.0     45.0     25.0     25.0     25.0      8.2      8.2     12.3     12.3     12.3     12.3
24.0      3.1     36.2     42.5     37.0     43.4      3.1   27653.74    389.9
   3      15.0     45.0     25.0     25.0     25.0      3.0      3.0     12.3     12.3     12.3     12.3
20.7      3.0     30.9     32.4     33.2     33.2      3.0   23855.28    346.6
   4      15.0     44.6     25.0     25.0     25.0      3.0      3.0     12.3     12.3     12.3     12.3
14.9      3.0     21.1     23.2     24.1     24.1      3.0   20386.21    303.3
```

Kasus 4

periode 1

generator	output mw	limit	inc cost $/mwhr	penalty fact	operating cost $/hr
UNIT1	15.0	max	40.5405	1.0000	558.21
UNIT2	45.0	max	76.7110	1.0000	2316.98
UNIT3	25.0	max	33.2410	1.0000	806.10
UNIT4	25.0	max	30.6445	1.0000	766.53
UNIT5	25.0	max	30.6445	1.0000	766.53
UNIT6	13.7		100.5353	1.0000	1917.66
UNIT7	13.7		100.5353	1.0000	1917.66
UNIT8	12.3	max	25.8302	1.0000	284.00
UNIT9	12.3	max	25.8302	1.0000	284.00
UNIT10	12.3	max	25.8302	1.0000	284.00
UNIT11	12.3	max	25.8302	1.0000	284.00
UNIT12	24.0	max	88.2493	1.0000	2314.47
UNIT13	6.4		100.5353	1.0000	1414.27
UNIT14	36.2	max	88.1572	1.0000	3277.25
UNIT15	45.0	max	94.5805	1.0000	3971.96
UNIT16	37.0	max	87.1034	1.0000	3214.70
UNIT17	45.0	max	93.8360	1.0000	3938.46
UNIT18	6.4		100.5353	1.0000	1414.27
totals	411.6				29731.07
lambda =	100.5353				

total load = 411.6 total losses = 0.0

periode 2

generator	output mw	limit	inc cost $/mwhr	penalty fact	operating cost $/hr
UNIT1	15.0	max	40.5405	1.0000	558.21
UNIT2	45.0	max	76.7110	1.0000	2316.98
UNIT3	25.0	max	33.2410	1.0000	806.10
UNIT4	25.0	max	30.6445	1.0000	766.53
UNIT5	25.0	max	30.6445	1.0000	766.53
UNIT6	8.7	min	93.1876	1.0000	1433.36
UNIT7	8.1		92.2445	1.0000	1373.86
UNIT8	12.3	max	25.8302	1.0000	284.00
UNIT9	12.3	max	25.8302	1.0000	284.00
UNIT10	12.3	max	25.8302	1.0000	284.00
UNIT11	12.3	max	25.8302	1.0000	284.00
UNIT12	24.0	max	88.2493	1.0000	2314.47
UNIT13	3.1		92.2445	1.0000	1091.12
UNIT14	36.2	max	88.1572	1.0000	3277.25
UNIT15	42.2		92.2445	1.0000	3712.68
UNIT16	37.0	max	87.1034	1.0000	3214.70
UNIT17	43.1		92.2445	1.0000	3762.51
UNIT18	3.4	min	93.1164	1.0000	1123.80
totals	389.9				27654.10
lambda =	92.2445				

total load = 389.9 total losses = 0.0

periode 3

generator	output mw	limit	inc cost $/mwhr	penalty fact	operating cost $/hr
UNIT1	15.0	max	40.5405	1.0000	558.21
UNIT2	45.0	max	76.7110	1.0000	2316.98
UNIT3	25.0	max	33.2410	1.0000	806.10
UNIT4	25.0	max	30.6445	1.0000	766.53
UNIT5	25.0	max	30.6445	1.0000	766.53
UNIT6	3.7	min	85.8400	1.0000	985.79
UNIT7	3.0	min	84.8020	1.0000	925.52
UNIT8	12.3	max	25.8302	1.0000	284.00
UNIT9	12.3	max	25.8302	1.0000	284.00
UNIT10	12.3	max	25.8302	1.0000	284.00
UNIT11	12.3	max	25.8302	1.0000	284.00
UNIT12	20.6		83.8151	1.0000	2024.18
UNIT13	3.0	min	92.0946	1.0000	1085.53
UNIT14	30.7		83.8151	1.0000	2804.14
UNIT15	32.2	min	83.8288	1.0000	2832.31
UNIT16	33.1		83.8151	1.0000	2880.79
UNIT17	33.1	min	83.8288	1.0000	2882.15
UNIT18	3.0	min	92.0946	1.0000	1085.53
totals	346.6				23856.30
lambda =	83.8151				

total load = 346.6 total losses = 0.0

periode 4

```
generator  output   limit  inc cost penalty fact  operating cost
             mw              $/mwhr                   $/hr
---------  ------   -----  --------  ------------  --------------
    UNIT1   15.0     max     40.5405      1.0000           558.21
    UNIT2   44.2             75.7696      1.0000          2257.45
    UNIT3   25.0     max     33.2410      1.0000           806.10
    UNIT4   25.0     max     30.6445      1.0000           766.53
    UNIT5   25.0     max     30.6445      1.0000           766.53
    UNIT6    3.0     min     84.8020      1.0000           925.52
    UNIT7    3.0     min     84.8020      1.0000           925.52
    UNIT8   12.3     max     25.8302      1.0000           284.00
    UNIT9   12.3     max     25.8302      1.0000           284.00
   UNIT10   12.3     max     25.8302      1.0000           284.00
   UNIT11   12.3     max     25.8302      1.0000           284.00
   UNIT12   14.5             75.7696      1.0000          1535.68
   UNIT13    3.0     min     92.0946      1.0000          1085.53
   UNIT14   23.7     min     78.2912      1.0000          2236.76
   UNIT15   22.6             75.7696      1.0000          2068.13
   UNIT16   23.5             75.7696      1.0000          2117.97
   UNIT17   23.5             75.7696      1.0000          2117.97
   UNIT18    3.0     min     92.0946      1.0000          1085.53
---------  ------                                    --------------
 totals     303.3                                        20389.45
 lambda =    75.7696

total load =      303.3  total losses =        0.0
```

final output

```
FINAL OUTPUT DYNAMIC ECONOMIC DISPATCH

PERIOD       UNIT GENERATION
PCOST     LOAD
             1       2       3       4       5       6       7       8       9      10      11
12       13      14      15      16      17      18          R/HR      MW
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
   1       15.0    45.0    25.0    25.0    25.0    13.7    13.7    12.3    12.3    12.3    12.3
24.0     6.4    36.2    45.0    37.0    45.0     6.4   29731.07    411.6
   2       15.0    45.0    25.0    25.0    25.0     8.7     8.1    12.3    12.3    12.3    12.3
24.0     3.1    36.2    42.2    37.0    43.1     3.4   27654.10    389.9
   3       15.0    45.0    25.0    25.0    25.0     3.7     3.0    12.3    12.3    12.3    12.3
20.6     3.0    30.7    32.2    33.1    33.1     3.0   23856.30    346.6
   4       15.0    44.2    25.0    25.0    25.0     3.0     3.0    12.3    12.3    12.3    12.3
14.5     3.0    23.7    22.6    23.5    23.5     3.0   20389.45    303.3
```

Kasus 5

periode 1

```
generator  output   limit  inc cost penalty fact  operating cost
             mw              $/mwhr                   $/hr
---------  ------   -----  --------  ------------  --------------
    UNIT1   15.0     max     40.5405      1.0000           558.21
    UNIT2   42.8     max     73.9983      1.0000          2147.44
    UNIT3   22.0     max     31.9555      1.0000           708.30
    UNIT4   22.0     max     30.1775      1.0000           675.29
    UNIT5   22.0     max     30.1775      1.0000           675.29
    UNIT6   14.8     max    102.0690      1.0000          2023.39
    UNIT7   14.8     max    102.0690      1.0000          2023.39
    UNIT8   11.1     max     24.5667      1.0000           253.06
    UNIT9   11.1     max     24.5667      1.0000           253.06
   UNIT10   11.1     max     24.5667      1.0000           253.06
   UNIT11   11.1     max     24.5667      1.0000           253.06
   UNIT12   24.0     max     88.2493      1.0000          2314.47
   UNIT13   15.7             123.5024      1.0000          2454.63
   UNIT14   36.2     max     88.1572      1.0000          3277.25
   UNIT15   42.8     max     92.6869      1.0000          3761.28
   UNIT16   37.0     max     87.1034      1.0000          3214.70
   UNIT17   42.8     max     91.9425      1.0000          3729.46
   UNIT18   15.7             123.5024      1.0000          2454.63
---------  ------                                    --------------
 totals     411.6                                        31029.97
 lambda =   123.5024

total load =      411.6  total losses =        0.0
```

periode 2

```
generator  output   limit  inc cost penalty fact  operating cost
             mw              $/mwhr                   $/hr
---------  ------   -----  --------  ------------  --------------
    UNIT1   15.0     max     40.5405      1.0000           558.21
    UNIT2   42.8     max     73.9983      1.0000          2147.44
    UNIT3   22.0     max     31.9555      1.0000           708.30
    UNIT4   22.0     max     30.1775      1.0000           675.29
    UNIT5   22.0     max     30.1775      1.0000           675.29
```

UNIT6	10.4		95.7051	1.0000	1595.16
UNIT7	10.4		95.7051	1.0000	1595.16
UNIT8	11.1	max	24.5667	1.0000	253.06
UNIT9	11.1	max	24.5667	1.0000	253.06
UNIT10	11.1	max	24.5667	1.0000	253.06
UNIT11	11.1	max	24.5667	1.0000	253.06
UNIT12	24.0	max	88.2493	1.0000	2314.47
UNIT13	5.7	min	98.7729	1.0000	1343.25
UNIT14	36.2	max	88.1572	1.0000	3277.25
UNIT15	42.8	max	92.6869	1.0000	3761.28
UNIT16	37.0	max	87.1034	1.0000	3214.70
UNIT17	42.8	max	91.9425	1.0000	3729.46
UNIT18	12.7	min	116.0835	1.0000	2095.25
totals	389.9				28702.75
lambda =	95.7051				

total load = 389.9 total losses = 0.0

periode 3

generator	output mw	limit	inc cost $/mwhr	penalty fact	operating cost $/hr
UNIT1	15.0	max	40.5405	1.0000	558.21
UNIT2	42.8	max	73.9983	1.0000	2147.44
UNIT3	22.0	max	31.9555	1.0000	708.30
UNIT4	22.0	max	30.1775	1.0000	675.29
UNIT5	22.0	max	30.1775	1.0000	675.29
UNIT6	5.4	min	88.3575	1.0000	1135.00
UNIT7	3.2		85.1625	1.0000	946.37
UNIT8	11.1	max	24.5667	1.0000	253.06
UNIT9	11.1	max	24.5667	1.0000	253.06
UNIT10	11.1	max	24.5667	1.0000	253.06
UNIT11	11.1	max	24.5667	1.0000	253.06
UNIT12	21.7		85.1625	1.0000	2110.80
UNIT13	3.0	min	92.0946	1.0000	1085.53
UNIT14	32.4		85.1625	1.0000	2948.39
UNIT15	33.8		85.1625	1.0000	2966.22
UNIT16	34.7		85.1625	1.0000	3016.05
UNIT17	34.7		85.1625	1.0000	3016.05
UNIT18	9.7	min	108.6647	1.0000	1758.13
totals	346.6				24759.32
lambda =	85.1625				

total load = 346.6 total losses = 0.0

periode 4

generator	output mw	limit	inc cost $/mwhr	penalty fact	operating cost $/hr
UNIT1	15.0	max	40.5405	1.0000	558.21
UNIT2	42.8	max	73.9983	1.0000	2147.44
UNIT3	22.0	max	31.9555	1.0000	708.30
UNIT4	22.0	max	30.1775	1.0000	675.29
UNIT5	22.0	max	30.1775	1.0000	675.29
UNIT6	3.0	min	84.8020	1.0000	925.52
UNIT7	3.0	min	84.8020	1.0000	925.52
UNIT8	11.1	max	24.5667	1.0000	253.06
UNIT9	11.1	max	24.5667	1.0000	253.06
UNIT10	11.1	max	24.5667	1.0000	253.06
UNIT11	11.1	max	24.5667	1.0000	253.06
UNIT12	16.3		78.0752	1.0000	1670.63
UNIT13	3.0	min	92.0946	1.0000	1085.53
UNIT14	25.4	min	79.6385	1.0000	2371.58
UNIT15	25.4		78.0752	1.0000	2278.87
UNIT16	26.3		78.0752	1.0000	2328.71
UNIT17	26.3		78.0752	1.0000	2328.71
UNIT18	6.7	min	101.2458	1.0000	1443.26
totals	303.3				21135.12
lambda =	78.0752				

total load = 303.3 total losses = 0.0

final output

FINAL OUTPUT DYNAMIC ECONOMIC DISPATCH

PERIOD	UNIT GENERATION										
PCOST	LOAD										
	1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	R/HR	MW			
1	15.0	42.8	22.0	22.0	22.0	14.8	14.8	11.1	11.1	11.1	11.1
24.0	15.7	36.2	42.8	37.0	42.8	15.7	31029.97	411.6			

```
   2       15.0    42.8    22.0    22.0    22.0    10.4    10.4    11.1    11.1    11.1    11.1
24.0     5.7     36.2    42.8    37.0    42.8    12.7   28702.75    389.9
   3       15.0    42.8    22.0    22.0    22.0     5.4     3.2    11.1    11.1    11.1    11.1
21.7     3.0     32.4    33.8    34.7    34.7     9.7   24759.32    346.6
   4       15.0    42.8    22.0    22.0    22.0     3.0     3.0    11.1    11.1    11.1    11.1
16.3     3.0     25.4    25.4    26.3    26.3     6.7   21135.12    303.3
```

RIWAYAT HIDUP

Penulis lahir di Makassar tanggal 30 September tahun 1992. Pada masa kecilnya, penulis hidup berpindah-pindah. Penulis pernah tinggal di Ternate, Sidoarjo, Pekanbaru, Jakarta, dan Surabaya. Penulis menyelesaikan sekolah da sar nya di SD Pelita Jakarta Selatan. Kemudian penulis melanjutkan pendidikannya di SMP Negeri 41 Jakarta Selatan. Selanjutnya penulis menamatkan pendidikan nya di SMA Negeri 5 Surabaya. Saat ini penulis merupakan mahasiswa Jurusan Teknik Elektro Fakultas Teknologi Industri Institut Teknologi Sepuluh Nopember. Pada masa perkuliahan penulis aktif dalam kegiatan organisasi kemahasiswaan. Penulis pernah menjadi anggota Paduan Suara Mahasiswa ITS selama 2 tahun. Selain itu penulis juga pernah menjadi pengurus Badan Eksekutif Mahasiswa ITS sebagai staff Ditjen Badan Koordinasi Pemandu ITS Kementrian PSDM. Dan yang terakhir penulis pun pernah menjadi Koordinator Lomba Cipta Elektroteknik Nasional EE Event Jurusan Teknik Elektro ITS.